IMPLEMENTACIÓN DE RuGle PARA RESOLVER EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON VENTANAS DE TIEMPO DURAS (VRPHTW)

por

Mayra Xiomara Gamboa Ramírez
Octubre 07, 2016

Proyecto de grado presentado como requisito parcial
para optar al título de
Magister en Ciencias
(Ingeniería Industrial)
Universidad del Norte, Barranquila
2016

Comité:

Dr. Ruben D. Yie Pinedo (Director)

Aprobado por la División de Postgrados e Investigaciones en Ingeniería en cumplimiento de los requisitos exigidos para otorgar el título de Magíster en Ingeniería Industrial en área de énfasis en Gestión Industrial.

_____

Ing. Ruben Yie Pinedo., Ph. D.

Director del Proyecto

_____

Jurado

_____

Jurado

Barranquilla, Octubre 2016.

"Totus tuus ego sum et omnia mea tua sunt. Accipio te in mea omnia. ¡Praebe mihi cor tuum María!".

"Soy todo tuyo y todo lo mío es tuyo. Te recibo como mi todo. ¡Dame tu corazón, oh María!".

Todo por la Inmaculada, nada sin ella.

Jesús, José y María, os doy el corazón y el alma mía.

.

Jezu Ufam Tobie

Todo tuyo María

TABLA DE CONTENIDO

CAPÍTULOS

LISTA DE FIGURAS

Figura

LISTA DE TABLAS

Tabla

INTRODUCCIÓN

El Problema de Enrutamiento de Vehiculos ó VRP (Vehicle Routing Problem) aparece naturalmente como un problema central en transporte, distribución y logística. De acuerdo a **?**, este problema ha sido estudiado por más de medio siglo. El primer registro que se tiene en la literatura del enrutamiento de vehículos se encuentra en el artículo publicado por **?**, donde se buscaba resolver el Problema del Agente Viajero ó TSP (Travelling Salesman Problem), el cual es un caso particular del VRP. Pero no es sino hasta la publicación de **?**, donde se incorpora más de un vehículo a la formulación del problema, convirtiéndolo en el primer estudio existente en la literatura del VRP tal como se conoce en la actualidad. El primer artículo que llevó la frase: "enrutamiento de vehículo" en su título, se le atribuye a **?**. Otras versiones del VRP emergieron a principios de los 70´s.

Un gran volumen de publicaciones le precedieron a estos estudios del VRP, convirtiéndose, sin lugar a dudas, en uno de los problemas de optimización combinatoria más tratados en la literatura.

El clásico VRP consiste en determinar el mejor conjunto de rutas, para una flota de vehículos, que se originan y finalizan en un único depósito central, para la distribución de bienes a clientes geográficamente dispersos. Cada cliente debe ser atendido una sola vez por un único vehículo y su demanda satisfecha en una sola visita. Las demandas totales de los clientes que son atendidos por el mismo vehículo, no deben exceder la capacidad total del

mismo. El objetivo es minimizar el costo total de transporte, compuesto por el número de vehículos utilizados y el tiempo/distancia total recorrida, sujeto a las restricciones anteriores.

?, adiciona restricciones de ventanas de tiempo al clásico VRP, creándose así el Problema de Enrutamiento de Vehículos con Ventanas de Tiempo ó VRPTW (Vehicle Routing Problem With Time Windows), e introduce en el año de 1987 un banco de prueba, compuesto por 56 problemas de referencia (56 Benchmark Problems), conocido en la actualidad como Instancias de Solomon (ver: **Sección 3.2.1.1.**).

Según **?**, debido a la complejidad computacional y escasez de computadores, las investigaciones acerca del VRP en la década de los 80´s generaron diferentes configuraciones estáticas del problema original. Versiones estocásticas, dinámicas y difusas del VRP no fueron muy estudiadas durante este periodo. A finales de esta década se introdujo el término metaheurística, que apareció por primera vez en el artículo publicado por **?**, para nombrar a un número de algoritmos de búsqueda empleados para resolver estos problemas de enrutamiento de vehículos y otros problemas de optimización combinatoria. En la década de los 90´s, las investigaciones acerca del VRP se incrementaron, principalmente gracias a la capacidad y disponibilidad de computadores, lo que permitió que los investigadores desarrollaran e implementaran algoritmos de búsqueda cada vez más complejos. A medida que ha aumentado el poder de los computadores, los investigadores han sido capaces de resolver problemas de enrutamiento de vehículos cada vez más grandes y de forma más eficiente.

Debido al sinnúmero de aplicaciones reales e importancia económica, la presente investigación considera a la variante del clásico VRP que posee restricciones de tiempo fijadas de antemano sobre los períodos del día en que debe llevarse a cabo las entregas: el VRPTW. En este problema, los clientes tienen ventanas de tiempo asociadas que se caracterizan por el tiempo más temprano (límite inferior de la ventana de tiempo) y el tiempo más tarde (límite superior de la ventana de tiempo) dentro del cual se debe prestar el servicio. Estas ventanas de tiempo pueden ser suaves (VRPSTW: Vehicle Routing Problem with Soft Time Windows)

o duras (VRPHTW: Vehicle Routing Problem with Hard Time Windows). Suaves significa que es permitido la violación de la ventana de tiempo, pero asociada con una penalización. En contraste, las ventanas de tiempo duras no permiten ningún retraso.

De acuerdo a **?**, todos estos problemas de optimización combinatoria han demostrado ser NP-Hard, Non Deterministic Polynomial Time Hard - Tiempo Polinomial No Determinista Duro. Y aunque hay conocidas técnicas para resolver instancias muy pequeñas de este problema de forma exacta, como los métodos Branch and Bound (**?**); (**?**), Generación de Columnas (**?**) y Relajación de Lagrange (**?**), es mucho más extensa la literatura sobre técnicas heurísticas (**?**).

Según **?**, se ha encontrado que los estudios que tratan a la variante más popular del VRP, el VRPTW, y los problemas combinatorios en general, tienden a dirigirse hacia el campo de las metaheurísticas, es decir, heurísticas con habilidad de conducir a otras técnicas más allá de los simples óptimos locales y constituyen la frontera entre Investigación de Operaciones e Inteligencia Artificial (heurísticas de optimización inteligente). Entre ellas: Colonias de Hormigas (AC: Ant Colonies), Búsqueda Tabú (TS: Tabú Search) y Algoritmos Genéticos (GA: Genetic Algorithms), curiosamente inspiradas por fenómenos naturales (físicos, biológicos, etc.) y, por lo general, obtenidas de utilizar varios agentes de búsqueda (por ejemplo, hormigas, cromosomas, etc.) que operan en competencia o cooperación e incorporan adaptatividad para ajustar la configuración del método (por ejemplo, una memoria sensible) y para ajustar el modelo o representación del problema (por ejemplo, mediante una función de regresión).

Según **?**, en los años recientes, aproximaciones híbridas se han vuelto populares para resolver el VRPTW. De acuerdo a **?**, dichas cooperaciones o hibridaciones, inicialmente se realizaron entre varias meaheurísticas. Pero hoy en día, se proponen cada vez más esquemas de cooperación entre métodos exactos y metaheurísticas. Estas cooperaciones buscan generar resultados de calidad integrando lo mejor de diferentes algoritmos, con el fin de obtener un desempeño superior al de la sola metaheurística.

La presente investigación, parte del hecho de que hay funciones ya existentes en otro software (o infraestructura ya existente en otras plataformas) cuyo código puede ser reutilizado debido a que se sabe que ya ha sido probado y funciona correctamente. Para lo cual, se puede acudir al uso de las API que sirven de interfaz entre programas diferentes. En vista del gran potencial que esconde la plataforma online Google Maps, se pretende aprovechar ésta, con el fin de resolver problemas de enrutamiento de vehículos cada vez más cercanos a la realidad. A través de una herramienta informática, desarrollada con el lenguaje de programación Ruby y basada en la API de Google, con el fin de encontrar la solución al VRPHTW sometido a rutas con congestión en determinadas horas del día.

CAPÍTULO 1

RESUMEN EJECUTIVO

Los VRP y sus variantes, son unos de los problemas de optimización combinatoria más tratados en la literatura. Estos ya han demostrado ser NP-Hard, debido a que, a medida que aumenta el número de nodos en la red objeto de estudio, el espacio de soluciones a explorar se incrementa de manera desmesurada, por lo tanto, una búsqueda exhaustiva entre todo el espacio de soluciones posible no sería factible y aún no se conoce un algoritmo en Tiempo Polinomial P) para resolverlos de manera óptima.

La presente investigación pretende aprovechar el gran potencial que esconde la plataforma online Google Maps, con el fin de resolver la variante más tratada en la literatura del VRP: EL Problema de Enrutamiento de Vehículos con Ventanas de Tiempo Duras (VRPHTW), sometido a rutas con congestión en determinadas horas del día. A través de RuGle, una herramienta informática de código abierto, desarrollada en el lenguaje de programación Ruby y basada en la API de Google Maps, para resolver este problema combinatorio multiobjetivo. Para las instancias de prueba se utilizaron ubicaciones reales, y para calcular la distancia, tiempo de viaje con estimación de tráfico y sin ella, entre los nodos que conforman la red objeto de estudio (instancias), se empleó la API de Google Maps.

CAPÍTULO 2

INFORMACIÓN GENERAL DEL PROYECTO

Tabla 2.1: Información General del Proyecto

TÍTULO DE LA INVESTIGACIÓN		
IMPLEMENTACIÓN DE RuGle PARA RESOLVER EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON VENTANAS DE TIEMPO DURAS(VRPHTW)		
NOMBRE DEL ESTUDIANTE	**CÓDIGO**	**PROGRAMA**
Mayra Xiomara Gamboa Ramírez	22668417	Ingeniería Industrial

PROYECTO ADSCRITO AL GRUPO (S)	**DIRECTOR DEL PROYECTO**
Grupo de Investigación en Productividad y Competitividad	Ruben Yie Pinedo Ph.D.

PALABRAS CLAVE				**DURACIÓN**
VRPHTW Multiobjetivo	Script	Problema Combinatorio	Google Maps API	12 Meses

CAPÍTULO 3

DESCRIPCIÓN DEL PROYECTO

En este capítulo se presenta una descripción detallada del proyecto, que incluye el planteamiento de la pregunta de investigación y su justificación, los aportes realizados, el estado del arte y marco teórico, los objetivos que se pretenden alcanzar, la metodología empleada, al igual que los resultados esperados y beneficiarios potenciales del mismo, así como el impacto esperado.

3.1. PREGUNTA DE INVESTIGACIÓN Y SU JUSTIFICACIÓN

En esta sección, se presenta el planteamiento del problema y su justificación, al igual que los aportes del presente proyecto, esta información le será de gran utilidad al lector para comprender las razones que llevaron a desarrollar esta investigación y cuales fueron las principales contribuciones de la misma.

3.1.1. Planteamiento de la Pregunta o Problema de Investigación

De acuerdo a **?**, un problema de enrutamiento simple es aquel cuyo único objetivo es encontrar la ruta mínima entre varios nodos geográficamente dispersos. Por otro lado, los problemas de enrutamiento y scheduling son mucho más complejos que los de enrutamiento puro, ya que no sólo deben encontrar la ruta mínima, sino que además deben solucionar aspectos como la utilización adecuada de los recursos (vehículos), lo cual, sumado a las restricciones típicas del problema, hacen muy difícil y costoso encontrar la solución óptima.

Estos problemas de optimización combinatoria, demandan enormes requerimientos computacionales, debido a que las soluciones se crean por medio de diferentes combinaciones posibles en las que todos los nodos que conforman la instancia a estudiar pueden ser visitados, y este número de combinaciones es de orden factorial. A medida que se incrementa el número de elementos a seleccionar, el espacio de las soluciones factibles a estos problemas crece de manera desmesurada, dificultando la obtención de la mejor solución (óptimo global) en un tiempo polinómico.

En el caso del TSP (Travelling Salesman Problem), que es el más simple y de enrutamiento puro, para una instancia de n nodos, el número de combinaciones posibles a explorar sería de $(n-1)!/2$. Si se tuviese, por ejemplo, un problema de 3 nodos, existiría una única solución, mientras que para un problema de 50 nodos, el número de soluciones a explorar pasaría a ser de 3×10^{62}. Por lo tanto, se hace evidente el hecho de que son problemas muy exigentes en cálculo y que no es factible realizar una búsqueda exhaustiva entre todas las soluciones posibles en problemas de tamaño considerable, por lo tanto, sólo instancias relativamente pequeñas pueden ser resueltas hasta la optimalidad.

Debido a que el VRP es NP-Hard, por restricción, el VRPTW también es NP-Hard. De hecho, incluso encontrando una solución factible al VRPTW cuando se tiene un número fijo de vehículos este problema resulta ser NP-Complete. Éste es un corolario del resultado

derivado por **?**, para el caso de un vehículo sin restricciones de capacidad, es decir, el TSPTW (Travelling Salesman Problem Whit Time Windows).

En el VRPHTW, un vehículo puede llegar a la locación del cliente antes del límite inferior de la ventana de tiempo a_i, pero debe esperar hasta que se abra la ventana de tiempo inferior para poder iniciar el servicio a ese cliente. Cualquier visita o servicio después del límite superior de la ventana de tiempo b_i produce soluciones infactibles. Esto evidencia las debilidades del VRPHTW al aplicarse a situaciones de la vida real, donde gestionar los vehículos para atender a todos los clientes dentro de las ventanas de tiempo establecidas por estos es muy difícil de lograr en la práctica.

Según (**?**), los proveedores logísticos pueden incrementar el número de vehículos, para responder rápidamente a la demanda de los clientes, con el fin de reducir el tiempo de espera de estos y cumplir con todas las ventanas de tiempo. Pero el inconveniente que surge es que, el costo de transporte es proporcional al tamaño de la flota vehicular, es decir que, a medida que se aumenta el número de vehículos de la flota, los costos asociados a ella también se van a acrecentar. Por lo tanto, cada vehículo adicional que se utilice causa un incremento en los costos fijos (compra de vehículos, costos de seguro y mantenimiento) y requiere un conductor adicional (salario de los conductores). El costo por distancia total recorrida es también un factor importante, porqué los precios del combustible son elevados constantemente y la contaminación se ha vuelto un tema crucial. Según **?**, a causa de su alto grado de industrialización y actividad económica, los transportes que transcurren en los países desarrollados son responsable del 30 % al 90 % del total de los gases contaminantes emitidos por el tráfico en todo el mundo.

En cuanto al tiempo de espera, éste afecta igualmente al proveedor logístico. Mucho tiempo de inactividad del vehículo en la locación del cliente o en un embotellamiento debido a una ruta muy congestionada, no solamente causa la pérdida de oportunidades para generar más ganancias sino que también hace que se incurra en costos extra, tales como: costos

operativos por vehículo, costos de mantenimiento y costos por tarifas de estacionamiento. Además de convertirse en un factor que contribuye al tráfico y a la contaminación medioambiental producida por el vehículo: congestión ocasionada por esperar en lugares inapropiados o por tomar rutas que de por sí ya están saturadas, polución del aire si espera con el motor encendido, entre otros.

Entonces, podríamos formular nuestra pregunta de investigación como: ¿Cómo minimizar los costos y tiempos de transporte en el VRPHTW?

Este problema es complejo y no puede resolverse efectiva y eficientemente basado sólo en la experiencia. Además de la complejidad debido a las ventanas de entrega fijadas de antemano por el cliente, sumado a las otras restricciones del modelo, el problema se torna aún más complicado cuando las rutas que conectan a los nodos que conforman la red objeto de estudio presentan una mayor congestión vehicular a ciertas horas del día. Por consiguiente, se vuelve prioritario desarrollar dinámicamente las rutas económicas de transporte para cada vehículo que atiende a este grupo de clientes geográficamente dispersos.

Tener en cuenta el tráfico, permite cumplir de manera más efectiva las solicitudes del cliente, puesto que al tomar una ruta considerando únicamente la distancia de un nodo a otro, no es garantía de llegar a tiempo a esa ubicación. Por lo tanto, es importante contar con información respecto a los horarios de mayor congestión de las rutas, con el fin de eliminar posibles retrasos, brindándole así mayor confiabilidad a los proveedores logísticos. Por otro lado, al evitar vías congestionadas es posible disminuir el tiempo de viaje, pero la distancia total recorrida por la flota podría incrementarse.

La presente investigación busca proponer una alternativa informática para resolver el problema de enrutamiento de vehículos con ventanas de tiempo duras (VRPHTW) sometido a rutas con congestión vehicular, persiguiendo los siguientes objetivos principales:

- minimizar el costo total de transporte, compuesto por: el número total de vehículos que conforman la flota vehicular y la distancia total recorrida por los vehículos de dicha flota;
- minimizar el tiempo total de espera de la flota vehicular (debido a llegadas anticipadas a la ubicación del cliente y a rutas con congestión).

Pero, para lograr esto, se debe encontrar la manera de calcular el estado de los arcos (vías) en diferentes horarios, con el fin de conocer de antemano los estimados de los tiempos de viaje con congestión, y sin ella, entre los diferentes nodos que conforman la red objeto de estudio (instancia), sin necesidad de incorporar ecuaciones para calcular el flujo vehicular que, sin lugar a dudas, incrementarían aún más la complejidad del modelo del VRPHTW a resolver. Fue así como se decidió hacer uso de la API de Google, la cual proporciona toda la información necesaria de geolocalización, direcciones, vías, etc., para alimentar el modelo y sin necesidad de acudir a ecuaciones de flujo vehicular para el cálculo de la congestión, simplemente haciendo consultas a través de esta interfaz y sin alterar la complejidad del problema original.

3.1.2. Justificación

Según **?**, el transporte es un dominio importante de la actividad humana: soporta y hace posible muchas actividades sociales y económicas. Cuando se utiliza, por ejemplo, el teléfono, se compra en un supermercado o almacén de alimentos, se lee el correo, o se viaja en una aerolínea por negocios o placer, se está siendo beneficiario de algún sistema que enruta mensajes, mercancías o personas de un lugar a otro. El transporte de carga, en particular, es una de las actividades más importantes de la actualidad.

Según **?**, **?**, el volumen de negocios de transporte de mercancías en Europa es de unos $168 billones por año. En el Reino Unido, Francia y Dinamarca, por ejemplo, el transporte

representa un 15 %, 9 %, y 15 % del gasto nacional, respectivamente. De acuerdo a **?**, **?**, se estima que los costos de distribución representan casi la mitad de los costos totales de logística y en algunas industrias, tales como la de alimentos y bebidas, los costos de distribución pueden representar hasta el 70 % del costo del valor agregado de la mercancía. (**?**) reporta que en 1989, 76.5 % de todo el transporte de mercancías se hizo a través de vehículos, lo que resalta la importancia de los problemas de enrutamiento y programación.

De acuerdo a **?**, el enrutamiento de vehículos es una actividad importante tanto en el sector público como en el sector privado. Por ejemplo, en los Estados Unidos, el costo de transporte representa aproximadamente el 10 % del producto interno bruto. En consecuencia, según **?**, incluso pequeñas mejoras en la eficiencia del enrutamiento puede dar lugar a grandes reducciones de costos. La mejora de la eficiencia del enrutamiento vuelve aún más importante a medida que los precios del combustible aumentan.

Según **?**, en años recientes, el transporte urbano de mercancías se ha convertido en un tema relevante en la planificación de las urbes, principalmente, porqué éste puede generar algunos problemas graves en las zonas metropolitanas, afectando, entre otras cosas, las ganancias de los distribuidores de logística, por lo tanto, herramientas eficaces para optimizarlo, hoy más que nunca, son requeridas. Y una de dichas herramientas es el VRP.

En cuanto a la tasa de crecimiento de la literatura que trata a este problema, **?**, encontraron que es casi perfectamente exponencial con un 6.09 % anual. Este hecho sólo demuestra la vitalidad del VRP. Gracias también a que en las últimas décadas, a medida que se ha incrementado el poder de los computadores, los investigadores han sido capaces de resolver VRP cada vez más grandes y de forma más eficiente. Sin embargo, esta tasa de crecimiento no es tan rápida si se compara con el de otras disciplinas contemporáneas de la Investigación de Operaciones y Ciencias de la Administración OR/MS (Operations Research/Management Science), por ejemplo: Análisis Envolvente de Datos DEA (Data Envelopment Analysis) con una tasa de 25.5 %, Flowshop Scheduling con 15.1 %, Cell Manufacturing con 10.6 % (**?**) y

Mass Customization con 10.6 % (**?**). El coeficiente de incremento más lento puede ser explicado de varias maneras: quizás soluciones del VRP requieren más sofisticación que las de los otros problemas de OR/MS. Este es ciertamente el caso con DEA y Mass Customization.

Se encontró también que, debido a la importancia económica del transporte, específicamente en la cadena de suministros, la eficiencia de la logística se torna crítica debido a factores que afectan:

1. Los Productos: principalmente por los ciclos de vida de los mismos, los cuales, en algunos casos pueden ser muy cortos,
2. Los Clientes: debido a los requerimientos cada vez más exigentes de los clientes (por ejemplo: ventanas de tiempo muy estrechas en la que aceptan ser atendidos por su distribuidor $[a_i, b_i]$), quienes demandan una respuesta rápida con tiempos de entrega cada vez más estrictos.
3. Al Proveedor Logístico: los costos de transporte, utilidades e imagen de la compañia ante sus clientes y la sociedad en general.

En cuanto a los productos, en el caso de alimentos perecederos o productos con ciclos de vida cortos, de apenas unas horas, es muy importante cumplir con los tiempos de entrega establecidos por el cliente, es decir, cumplir con las ventanas de tiempo pactadas, ya que las entregas tardías, entregas después del límite superior de la ventana de tiempo b_i, podrían afectar la calidad del producto, como por ejemplo, en el caso del hormigón (concreto): el concreto premezclado que se entrega en un camión mezclador, mantiene al producto en forma homogénea hasta que es vaciado en el lugar de colocación. Dicho producto permanece en estado plástico por varias horas, según el tipo de mezcla y condiciones de colocación. El concreto normalmente endurece entre dos y doce horas después del mezclado. Si el camión tiene que esperar durante un extenso período de tiempo antes de descargar el concreto, la calidad del producto puede verse afectada negativamente.

En cuanto al cliente, este espera un servicio de calidad, y pacta una ventana de tiempo en la que espera ser atendido, ni antes ni después. La satisfacción del cliente debe ser una prioridad si la organización desea permanecer en el mercado, mantener y aumentar sus ventas.

También podría encontrarse el caso en el que el proveedor logístico pretenda adelantar una entrega, antes del límite inferior de la ventana de tiempo a_i, o retrasar una entrega, después del límite superior de la ventana de tiempo b_i, encontrándose, por ejemplo, con que no hay nadie en la bodega del cliente que pueda recibir el pedido, por lo que resultaría imposible realizar la entrega antes ó después del tiempo pactado con el cliente.

Como se vio anteriormente, el incumplimiento de estas ventanas puede acarrear no solamente pérdidas cuantificables en el valor del dinero, tanto para el cliente como para el proveedor logístico, sino también pérdida de la estima y confianza en la compañia distribuidora, algo díficil de cuantificar, lo cual se verá reflejado negativamente en la imagen y utilidades futuras de la misma.

En la literatura se encontró que los problemas de enrutamiento son usualmente configurados con un sólo objetivo, generalmente el de minimizar costos. Otro objetivo importante para este tipo de problemas, y que además influye en los costos, pero que muchas veces es considerado como un objetivo secundario, es el tiempo total de espera.

Pero, aún cuando se halló que los problemas de enrutamiento han sido ampliamente estudiados, son usualmente configurados con un sólo objetivo. Sin embargo, la mayoría de problemas que se encuentran en la industria, particularmente en logística, presentan múltiples objetivos por naturaleza, los cuales frecuentemente entran en conflicto. Por esta razón adoptar un punto de vista multiobjetivo podría resultar ventajoso para este tipo de casos.

De igual forma, el campo de la optimización multiobjetivo está llamando cada vez más la atención de investigadores, principalmente porqué ofrece novedosas oportunidades para la definición de problemas, no obstante, la relativamente moderada tasa de publicaciones pare-

ciese sugerir que aún continúa siendo muy joven el dominio de los problemas de enrutamiento multiobjetivo, por lo que aún hay mucho por desarrollar en esta área.

Con todo lo anterior, se corroboró la relevancia de esta investigación, tanto desde el punto de vista teórico, como desde el punto de vista práctico. Al igual que la necesidad de explorar y desarrollar estrategias alternativas para resolver problemas multiobjetivo de enrutamiento de vehículos, teniendo en cuenta aspectos que los hagan más realistas, como el hecho de considerar los tiempos de entrega fijados de antemano por los clientes y las rutas con congestión vehicular.

3.1.3. Aportes

Esta investigación, pretende brindar una herramienta informática denominada RuGle (Ruby + API Google), la cual puede ser de gran utilidad para todos aquellos que se inician por primera vez en el estudio del Problema de Enrutamiento de Vehículos con Ventanas de Tiempo Duras. La herramienta propuesta, utiliza como estrategia de solución a la clásica heurística de inserción I1 de **?**. Ésta heurística se modificó para poder interactuar y aprovechar las ventajas de la plataforma online de Google Maps y se codificó en el lenguaje de programación de código abierto Ruby, permitiéndole al usuario ver y modificar el código fuente, con el fin de incluir otras estrategias de búsqueda, construcción y mejoramiento de soluciones para el VRPHTW partiendo del código de RuGle. Esta herramienta no utiliza los famosos Bancos de Prueba para el VRPHTW empleados por diferentes autores, y presentados en la **Sección 3.2.1.1.**, en cambio, le permite al usuario construir sus propias instancias de prueba a partir de ubicaciones reales en cualquier lugar del mundo y con una estructura muy similar, en la medida de lo posible, a las propuestas por **?** y **?**, tal como se describe en el Manual del Usuario de RuGle **(Apendice B)**. Todo esto, con el fin de afianzar los conceptos básicos de un VRPHTW, hallar su solución, y ser un punto de partida para apro-

vechar el gran potencial que esconde la plataforma online de Google Maps en el estudio y planteamiento de VRPHTW cada vez más complejos.

A continuación, se presentan las diferencias entre RuGle y otras aplicaciones de la Heurística de Inserción I1:

- RuGle no utiliza la distancia euclidiana, en cambio utiliza la distancia real entre dos ubicaciones en cualquier lugar del mundo, empleando para ello la API de Google Maps.
- RuGle incluye el componente de la congestión en determinadas horas del día, el cual no es tenido en cuenta en muchos modelos del VRPHTW, y sin incrementar la complejidad del problema original, gracias también al API de Google Maps.
- Rugle siempre busca el mejor instante posible en toda la línea de tiempo que hay entre un nodo i y un nodo j para la construcción de las rutas. La herramienta hace la petición al servidor de Google y finalmente toma el que arroja un menor tiempo de viaje entre ellos.

RuGle en vez de calcular la distancia, tiempo y tiempo con estimación de tráfico entre los nodos que componen la red objeto de estudio, simplemente hace una petición al API de Google Maps, la cual envía información respecto al origen, destino, modo (para éste caso el modo es conduciendo), hora de salida y el modelo de tráfico (para éste caso el modelo de tráfico se considera cómo pesimista), dicha petición se hace teniendo en cuenta la hora en la cual el vehículo saldrá del nodo origen para atender a los clientes, obteniendo así un enfoque más cercano a la realidad.

Otro de los aportes que esta investigación realiza a todos aquellos que se inician en el campo del VRPHTW, es explicar de manera sencilla los tiempos que componen este problema: Tiempo de Viaje, Tiempo de Llegada, Tiempo de Espera, Tiempo de Inicio de Servicio y Tiempo de Servicio. Y como calcularlos, así como ilustrar de forma didáctica los

tres casos que se pueden presentar al momento de resolver problemas de enrutamiento de vehículos con restricciones de ventanas de tiempo (ver: **Capítulo 4.3**). Entender esto es fundamental al momento de entrar a codificar una estrategia de solución para este tipo de problemas, debido a que una mala comprensión de los mismos llevará a la construcción de soluciones infactibles. Finalmente, se brinda una herramienta autodidáctica de código abierto que ayudará a afianzar los conceptos básicos de un VRPHTW multiobjetivo y su solución. Para ampliar la información de esta sección, consultar el **Capítulo 4**.

3.2. ESTADO DEL ARTE Y MARCO TEÓRICO

El presente capítulo, presenta los antecedentes y las definiciones necesarias que fundamentan teóricamente en forma básica y sintetizada el presente trabajo.

3.2.1. Antecedentes

En la actualidad existen diferentes algoritmos propuestos para la solución del problema de enrutamiento de vehículos con ventanas de tiempo. A continuación se presentan la siguiente revisión literaria de los procedimientos para resolver el VRPTW:

De acuerdo a **?**, estudios previos del VRPTW incluyen ambos, algoritmos de optimización y aproximaciones heurísticas, pero investigaciones recientes se enfocan en estas últimas debido a la complejidad del VRPTW. Generalmente, las aproximaciones heurísticas pueden dividirse en dos áreas: Construcción de Rutas y Mejoramiento de Rutas. Los procedimientos de construcción inicial de la solución incluyen el método *Savings* (Ahorros) de **?**, el método *Nearest Neighbor* (Vecino más Cercano) y la heurística *I1* de **?**, (Solomon´s Route Construction Heuristic I1). Los procedimientos de mejora de búsqueda local incluyen métodos de Intercambio de Rama tales como la aproximación *Or-Opt* y métodos de Intercambio de Nodos tales como la aproximación *Swap* y la aproximación λ-*exchange* de **?**.

Para mejorar el procedimiento de búsqueda local, los investigadores están recurriendo a metaheurísticas para obtener soluciones superiores. Estas pueden ser mejores que el óptimo local y algunas veces incluso iguales a la solución óptima global. Metaheurísticas incluyen Búsqueda Tabú (TS: Tabú Search) de **?**, Aceptación por Umbrales (TA: Threshold Accepting) de **?** y el Método Noising (NM: Noising Method) de **?**.

Según **?**, para el correcto desempeño de muchos de estas técnicas heurísticas y metaheurísticas es importante la definición de movimientos, también denominados entornos o vecin-

darios, que permitan pasar de una solución a otra cercana en el espacio de búsqueda. Estos movimientos vecinales deben poseer las siguientes características: cada vecindario debe contener un gran número de soluciones vecinas que sean fáciles o rápidas de evaluar. De otra forma la exploración del espacio de soluciones puede no ser eficaz. Tradicionalmente en los problemas de rutas de vehículos los movimientos vecinales pueden ser "intrarutas" o "entrerutas". Cuando se habla de movimientos vecinales "intrarutas" se refiere a las modificaciones de cada ruta independiente de las demás (cambio de orden de los nodos de visita de esa ruta). Entre ellos se destacan los conocidos intercambios *r-óptimos* **?**; **?**. En cuanto a los movimientos vecinales "entrerutas", estos se refieren a las modificaciones que afectan a más de una ruta (pasar elementos de una ruta a otra, o intercambio de elementos entre rutas). Entre ellos se encuentran los propuestos por **?**, o **?** que generalizan las usadas anteriormente por **?**.

Algunos investigadores están recurriendo a un mecanismo basado en el método de "Ejection Chains" ("Cadenas de Expulsiones") concebido por **?**, en el contexto del TSP, para generar soluciones de forma diferente. De acuerdo a **?**, las características básicas que distinguen a los métodos basados en Ejection Chains de los movimientos clásicos son: 1) El movimiento de una solución a otra no es simple, sino que antes de realizarse el cambio se genera una cadena de movimientos y la nueva solución se elige entre las que aparecen en esa cadena. 2) Estas cadenas operan directamente sobre estructuras similares a una solución que se denominan "estructuras de referencia", más no sobre soluciones como tal. Para generar estas cadenas de movimientos se debe disponer de una serie de reglas para crear estas estructuras desde una solución, para pasar de una estructura de referencia a otra (reglas de transición), y para crear soluciones factibles a partir de una estructura de referencia. Según (Glover, 1996), los Procedimientos Ejection Chains se basan en la idea de generar secuencias compuestas de movimientos o cadenas de movimiento, que conducen de una solución a otra, a través de pasos vinculados en los que cambios en los elementos seleccionados causan la "expulsión" de otros elementos de su posición actual. Estas cadenas pueden contener cantidades

exponenciales de soluciones, pero cuyo mejor elemento puede identificarse en tiempo polinomial. Implementaciones de Ejection Chains han producido buenos resultados al momento de resolver problemas de enrutamiento de vehículos, incluso con complicadas restricciones como en (**?**), (**?**), (**?**), (**?**) y (**?**).

Según **?**, recientemente, las aproximaciones híbridas se han vuelto populares. Por ejemplo, **?** emplean Búsqueda Tabú (TS: Tabu Search) y Algoritmo Genético (GA: Genetic Algorithm) para resolver el VRPHTW, encontrándose un desempeño superior al de la sola metaheurística. **?** combinan Algoritmo Genético, Recocido Simulado (SA: Simulated Annealing) y Búsqueda Tabú en la previsión de mantenimiento. **?** integran Búsqueda Tabú con Recocido Simulado en varias aplicaciones. **?** propone un algoritmo de Formación de Umbrales Tabú, que emplea Búsqueda Tabú como base y el valor umbral para limitar el número de búsquedas. El demostró que la aproximación híbrida puede generar resultados de calidad porque integra lo mejor de diferentes algoritmos. **?** combinan Estrategias de Evolución y Búsqueda Tabú para minimizar el número de vehículos y la distancia total para el VRPTW. **?** propuso un algoritmo híbrido de dos etapas, el cual combina el Reocido Simulado y Búsqueda Grande de la Vecindad para el Pickup and Delivery VRPTW y Múltiples Vehículos (PDPTW). **?** resolvió el VRP Capacitado (CVRP) por un híbrido del Recocido Simulado basado en el Vecino más Cercano para minimizar el costo de la flota heterogénea y maximizar la utilización de capacidad.

? proponen una taxonomía para métodos de optimización hibrida que involucran aproximaciones heurísticas, con el fin de considerar esquemas cooperativos entre métodos exactos y metaheuríticos.

Se encontró que, a tráves de los años, un gran número de artículos de investigación han sido dedicados al VRPHTW. Por el contrario, en cuanto al VRPSTW, el número de referencias es relativamente pequeño. (**?**).

? presentan la siguiente revisión de la literatura que trata a esta variante del VRPTW:

? consideran un problema de enrutamiento Pickup-and-Delivery (Recogida y Entrega) con un sólo vehículo donde cada carga tiene su propio origen y su propio destino. Ellos minimizan una combinación lineal del tiempo de operación total del vehículo y la penalización total debido a la violación de cualquiera de las ventanas de tiempo, y resuelven el problema por la Descomposición de Bender. **?** considera un sistema de distribución de un sólo vehículo con un único depósito en el que la actividad diaria de distribución inicia y finaliza. Se incorpora al cliente las preferencias específicas de la ventana de tiempo en el proceso de modelado y propone un Enfoque Objetivo (Goal Approach) para resolver el problema. Los objetivos son minimizar el tiempo total de viaje y la desviación respecto a las preferencias de la ventana de tiempo. **?** proponen un enfoque heurístico para el VRPSTW el cual itera entre un problema de asignación generalizada para asignar los clientes a los vehículos y una serie de problemas de enrutamiento y planeación (Scheduling) definidos por los problemas del agente viajero con restricciones de ventanas de tiempo suaves (TSPSTW: Traveling Salesman Problem with Soft Time Windows). **?** propone tres heurísticas para el VRPSTW con flota homogénea basada en el Vecino más Cercano, Savings de **?** y las reglas del espacio-tiempo. El costo de violar las ventanas de tiempo se asume que es una función lineal del monto de violación de la ventana de tiempo. **?** proponen una heurística de TS para resolver un VRPSTW en el que la distancia total recorrida más la penalización total en la que se incurra por no prestar el servicio en la ventana de tiempo se reduce al mínimo. **?** se ocupa de un problema multi-ship Pickup-and-Delivery (Recogida y Entrega) con ventanas de tiempo suaves, en el cual, se imponen costos inconvenientes por atender a los clientes fuera de sus ventanas de tiempo correspondientes. La suma de los costos de transporte y los costos inconvenientes se minimizan mediante la resolución de un conjunto de problemas particionados cuyas variables son los horarios factibles.

Por otra parte, vale la pena resaltar que la mayoría de problemas de enrutamiento de la vida real tienen otros objetivos además de minimizar el tiempo/distancia total de viaje o el costo total de distribución. Por ejemplo, evitar la subutilización de mano de obra o capacidad de un vehículo, mientras se cumple con las preferencias de los clientes con respecto a la hora del día en que les gustaría ser atendidos. Esto da lugar a un VRP multiobjetivo. Desde hace algunos años, se ha prestado gran atención al desarrollo de metodologías para tratar con problemas de programación multiobjetivo, entre ellas: **?**, **?** y **?**. Entre las más populares sobresale el enfoque Goal Programming (GP): (**?**), (**?**) y (**?**). Subyacente al GP está el concepto de satisfacer los objetivos, es decir, buscar una solución que se aproxime lo más posible a las metas. Para este propósito se estable un nivel objetivo, el cual representa un nivel aceptable de logro para cada objetivo. Variables de desviación se definen con el fin de representar a la cantidad por la cual son sobrepasadas en número o están sobre el objetivo correspondiente. Por lo general, los modelos GP se clasifican en dos categorías diferentes. En la programación de metas ponderadas (Weighted Goal Programming), los pesos son asignados a las desviaciones de los niveles de objetivos, los cuales reflejan la clasificación del tomador de decisiones de la importancia de esta desviación. Luego, la penalización total es minimizada. En la programación de metas lexicográfica (Lexicographic or Preemptive Goal Programming) se asignan prioridades a las metas. A continuación, se minimizan los objetivos de uno en uno en orden de prioridad. En cada minimización sucesiva, mayor nivel de prioridad se mantiene los valores óptimos como se indica en minimizaciones previas.

? consideran un problema general de enrutamiento de vehículos con ventanas de tiempo suaves y duras, de tamaño medio, una flota heterogénea de vehículos y múltiples objetivos. Su propósito es investigar el uso de GP para modelar el problema y proponer un enfoque para resolverlo proporcionando una solución óptima en un tiempo computacional razonable en términos de aplicaciones reales. La inclusión de múltiples objetivos y restricciones específicas en la configuración general del VRP hace más complicado el modelo y resalta las dificultades con respecto al tiempo necesario para resolverlo de manera óptima con software

de optimización estándar. Sin embargo, por otra parte, estas características también pueden ser explotadas para llegar a procedimientos de solución que, a primera vista, no parecen muy eficientes. Teniendo esto en cuenta, ellos no proponen resolver directamente el modelo GP resultante con un software de propósito general, sino que proponen llevar a cabo un enfoque en dos etapas que llamaron enumeración seguido por optimización (EFBO: Enumeration Followed by Optimization). Este procedimiento consiste en primer lugar en enumerar todas las rutas factibles y calcular su desviación de los objetivos. En segundo lugar, el mejor conjunto de rutas, de acuerdo a los objetivos previamente establecidos son seleccionadas por la solución de un conjunto del problema particionado. Un enfoque similar ha sido adoptado en **?** para hacer frente a un problema Multi-Ship Pickup-and Delivery con ventanas de tiempo suaves. Este enfoque también ha sido adoptado en **?** para obtener una solución óptima a un problema real de distribución de compuestos. Las características especiales de los vehículos, los cuales son compartimentados, reducen el número de rutas factibles. Como se observa en ambos artículos, la eficiencia del procedimiento se encuentra en el número de rutas factibles que han de ser tratados.

3.2.1.1. Base Experimental para el VRPHTW

El banco de prueba propuesto por **?** es un referente importante en la literatura del VRPTW para comparar resultados obtenidos en éste tipo de problemas. Aunque los conjuntos de prueba que se plantean no tienen las restricciones de un problema real, se utilizan en la comunidad científica para validar los resultados obtenidos.

Más recientemente, **?** extendieron el banco de prueba de Solomon, presentando nuevas instancias con 200, 400, 600, 800 y 1000 clientes, diseñadas, en la medida de lo posible, de la misma manera que los conjuntos de prueba originales de 100 clientes.

Cabe resaltar que estos Bancos de Prueba, tanto de Solomon como el de Gehring y Homberger, son empleados para todas las variantes del VRPTW, incluyendo al VRPHTW. El Banco con los Mejores Resultados Publicados sólo muestra los relacionados con la Función de Costo, es decir: Distancia Total Recorrida y Número Total de Vehículos, no publican ningún otro objetivo.

Durante la presente investigación, se construyeron instancias de prueba utilizando ubicaciones reales (ver: **Apéndice B**, **Sección B.1**). Tratando de hacerlas muy similares en estructura a las propuestas por **?**, y **?**, estas últimas se describen continuación:

Instancias de Solomon para el VRPTW (Solomon Instances): El profesor Marius M. Solomon diseñó en 1983, 56 instancias de prueba para el VRPTW, cada una con 100 clientes. Estas fueron publicadas por primera vez en su artículo (**?**). El banco de prueba propuesto por Solomon agrupa instancias basadas en datos de algunos de los problemas utilizados por **?** para el problema de enrutamiento estándar. Y, además, refleja varios factores estructurales en enrutamiento de vehículos y planificación o scheduling, tales como: datos geográficos, número de clientes atendidos por cada vehículo, porcentajes de clientes con restricciones de tiempo, y flexibilidad de la ventana de tiempo (horizonte de planificación - scheduling - corto o largo). Los clientes tienen una distribución $[0, 100]^2$.

Las instancias se agrupan en 6 categorías (conjuntos de prueba) que se denotan como: R1, R2, C1, C2, RC1 y RC2. Cada uno de los conjuntos de prueba contiene entre 8 y 12 instancias (Ver **Tabla 3.1**). En cuanto a la distribución geográfica de los clientes, en R1 y R2 los datos geográficos se generaron aleatoriamente con una distribución uniforme (Ver **Figura** C.**1**). En los conjuntos de prueba C1 y C2 los clientes son agrupados por zonas o clusters (Ver **Figura** C.**2** y **Figura** C.**3** respectivamente), y finalmente en los conjuntos de prueba RC1 y RC2 algunos clientes son distribuidos uniformemente, mientras que otros son agrupados por zonas o clusters (Ver **Figura** C.**4**). En cuanto a los clientes atendidos por

cada vehículo, los conjuntos de prueba R1, C1 y RC1 tienen poca flexibilidad de scheduling (horizonte de planificación corto o ventanas de tiempo muy estrechas) permitiendo atender a pocos clientes en cada ruta (aproximadamente de 5 a 10 clientes), (Ver **Figura** C**.6)**. Por otro lado, los conjuntos de prueba R2, C2 y RC2 tienen más.

flexibilidad de scheduling (horizonte de planificación largo o ventanas de tiempo más amplias) permitiendo atender más de 30 clientes por vehículo. Esto hace que los problemas sean bastante difíciles de resolver de manera exacta.

En cuanto a la flota vehicular, se parte del hecho que para cada una de las 6 categorías en las que se divide el Banco de Prueba de Solomon se cuenta con una flota homogénea de 25 vehículos con capacidad conocida. Los conjuntos de prueba R1, C1 y RC1 cuentan con una flota con una capacidad fija de 200 unidades por vehículo permitiendo atender a pocos clientes en cada ruta. Por otro lado, los conjuntos de prueba R2, C2 y RC2 cuentan con una flota con mayor capacidad: R2 y RC2 cuentan con una capacidad fija de 1.000 unidades por vehículo, mientras que los Tipo C2 cuentan con una flota con una capacidad de 700 unidades por vehículo permitiendo atender más clientes por ruta (Ver **Tabla 3.1**).

Las ventanas de tiempo para los conjuntos de prueba C1 y C2 son generadas para permitir buenas, incluso hasta óptimas soluciones cluster por cluster

Las coordenadas de los clientes son exactamente iguales para los conjuntos de prueba de un tipo dado, a excepción del tipo C, es decir, son iguales para todos los problemas de Tipo R y para todos los problemas de Tipo RC. En cuanto a los conjunto de prueba de Tipo C, las coordenadas de los clientes de los conjuntos de prueba C1 y C2 no son iguales (Ver **Figura** C**.2** y **Figura** C**.3**). Las demandas para todos los problemas de Tipo C son iguales, estas varían en función del cliente, con los valores de 10, 20, 30, 40 ó 50 unidades, lo mismo que los tiempos de servicio para los clientes, que para este tipo de problemas es una constante con valor de 90 unidades de tiempo. Para los Tipo R las demandas son iguales, también varían en

función del cliente, con valores discretos que van de 1 a 41 unidades, los tiempos de servicio para los clientes que hacen parte de este tipo de problemas es de 10 unidades de tiempo, un valor constante para todos los clientes que conforman las instancias de Tipo R. En cuanto a los problemas de Tipo RC, el cual es una mezcla de clientes ubicados geográficamente en clusters y otros de forma aleatoria, de igual forma las demandas son

una mezcla de las de Tipo R y Tipo C. Para cada una de las instancias que conforman los problemas Tipo RC las demandas son iguales, también varían en función del cliente, los que se encuentran ubicados en clusters tienen una demanda con valores de 10, 20, 30, 40 ó 50 unidades, mientras que los que se encuentran ubicados aleatoriamente presentan una demanda discreta que va de 2 a 35 unidades, los tiempos de servicio para los clientes que hacen parte de este tipo de problemas es de 10 unidades de tiempo, un valor constante para todos los clientes que conforman las instancias de Tipo RC (Ver **Tabla 3.2**).

Por otra parte, los problemas cambian dependiendo de la ventana de tiempo: algunos tienen ventanas de tiempo muy restrictivas, mientras que las ventanas de tiempo de otros problemas apenas suponen ninguna restricción. En cuanto a la densidad de ventanas de tiempo, es decir, el porcentaje de clientes con ventanas de tiempo, existen problemas con un 25 %, 50 %, 75 % y 100 % de clientes con ventanas de tiempo.

Para cada una de las 56 instancias de prueba, se han creado instancias de menor dimensión considerando únicamente los primeros 25 o 50 clientes. Esto lleva a que el número total de problemas o instancias de prueba llegue hasta 168.

El tiempo de viaje entre dos clientes se asume, usualmente, que es igual a la distancia de viaje más el tiempo de servicio del cliente del cual se parte. Además, se asume que la flota es homogénea.

En cuanto a la nomenclatura empleada para denominar cada uno de los problemas específicos dentro de cada grupo particular, esta consiste en utilizar el nombre del grupo (por

ejemplo, C1) y añadirle el ordinal del problema concreto (por ejemplo, 01). Así, los problemas particulares tendrán denominaciones del tipo C101, R101, etc.

Y, adicionalmente, se le agrega el número de clientes al nombre del problema concreto, por ejemplo, R101.25, R101.50 y R101.100 indicando instancias pertenecientes al conjunto de prueba R1, ordinal del problema (01), con 25, 50 y 100 clientes respectivamente.

Las figuras siguientes se generaron con datos tomados de la página del profesor Marius M, Solomon, graficando las coordenadas X y Y que se encuentran registradas en cada una de las categorías en las que se divide el Banco de Prueba propuesto por Solomon: R1, R2, C1, C2, RC1 y RC2, obteniendo los siguientes resultados:

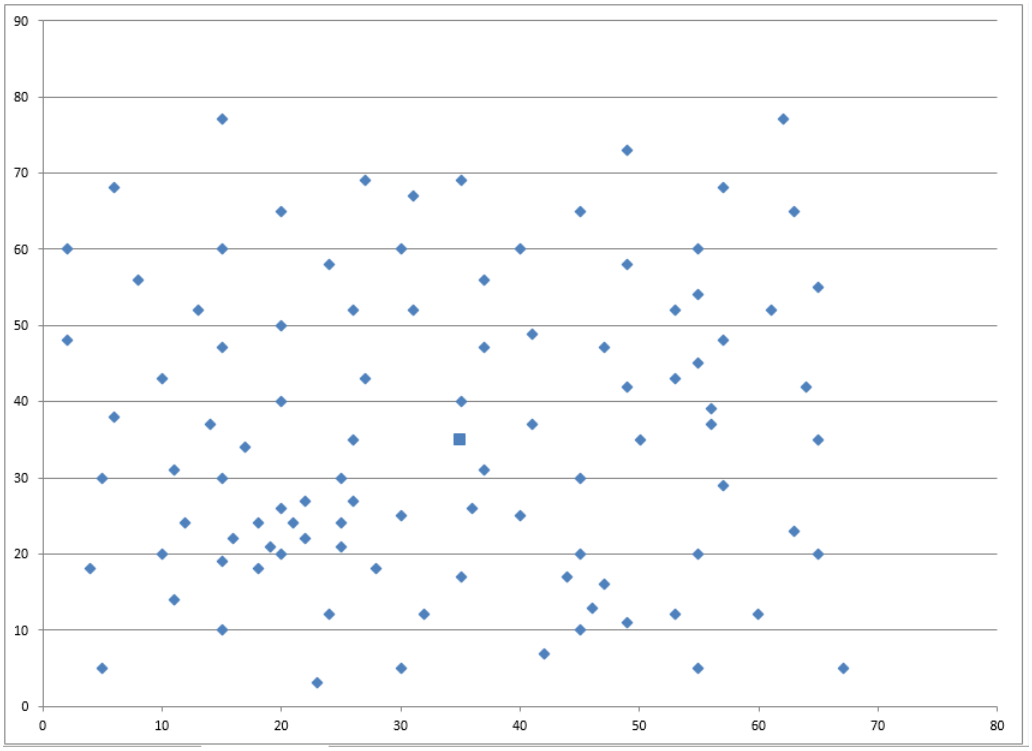

Figura 3.1: Distribución de los Clientes para los Conjuntos de Prueba R1 - R2

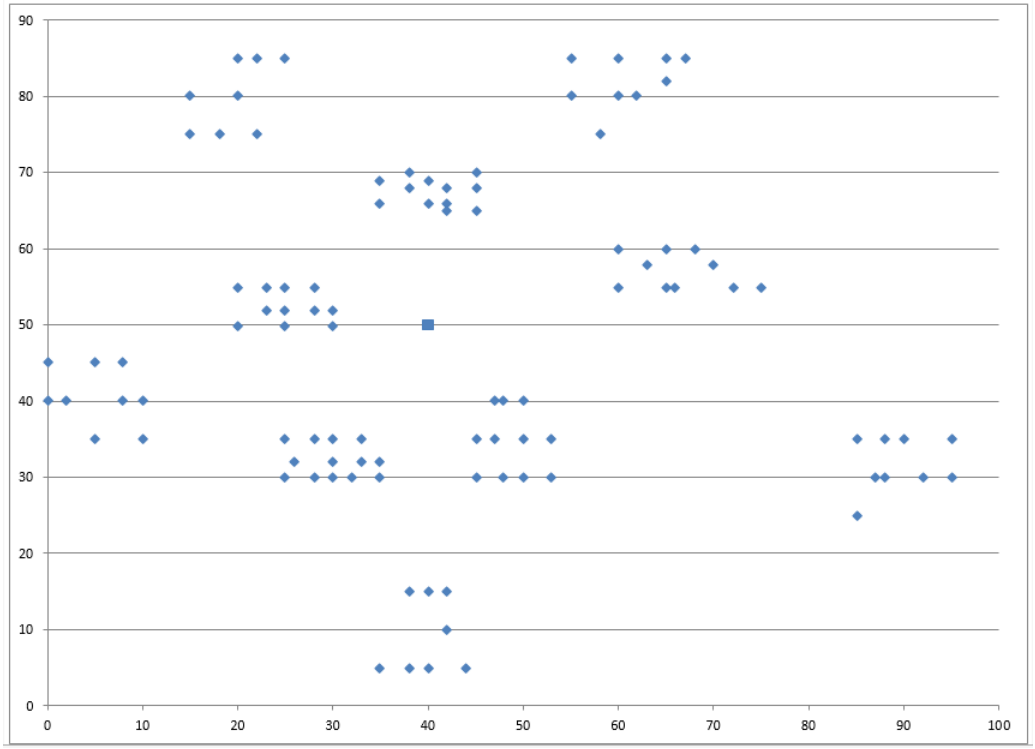

Figura 3.2: Distribución de los Clientes para los Conjuntos de Prueba C1

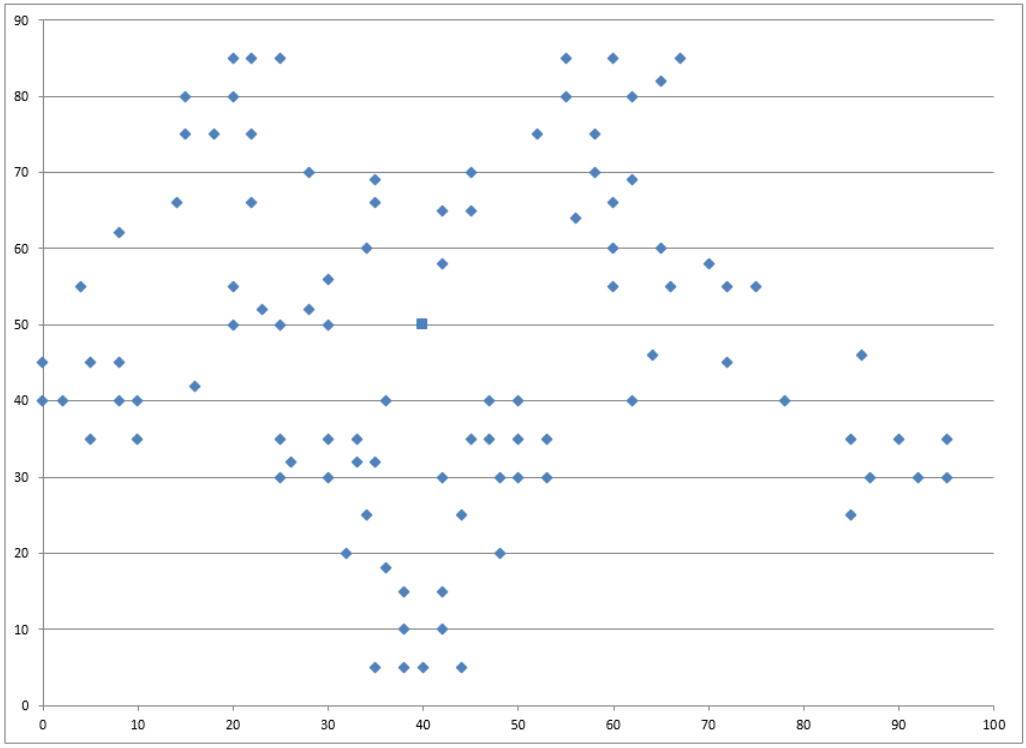

Figura 3.3: Distribución de los Clientes para los Conjuntos de Prueba C2

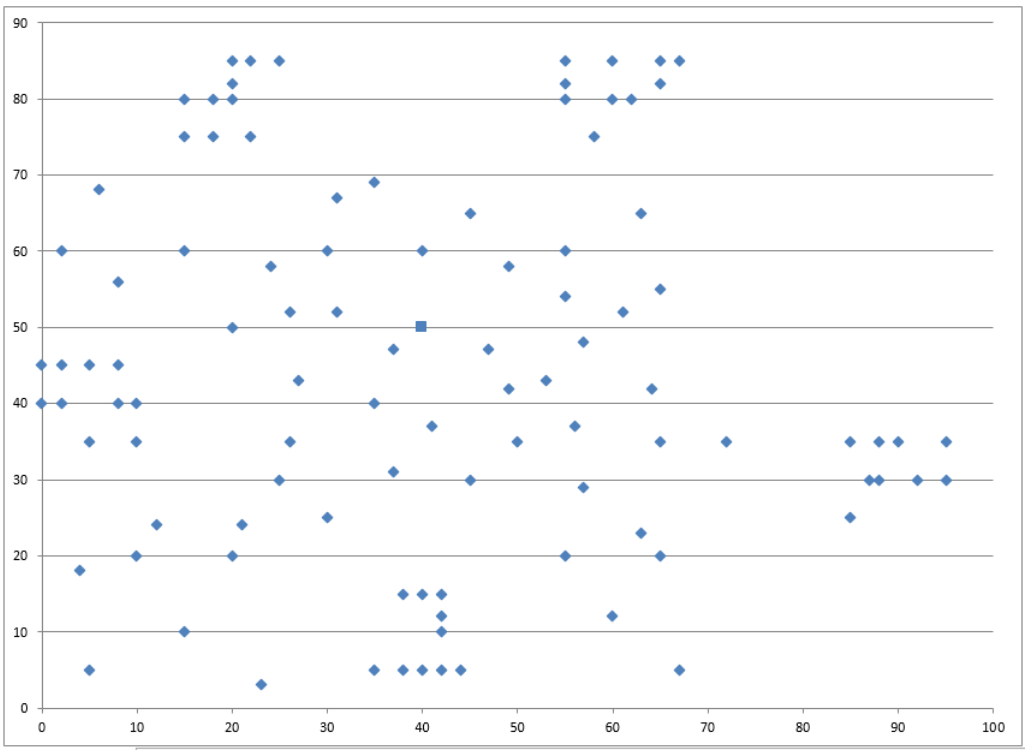

Figura 3.4: Distribución de los Clientes para los Conjuntos de Prueba RC1 - RC2

En la siguiente tabla se observan las 6 categorías en las que se divide el Banco de Prueba propuesto por el profesor Solomon: R1, R2, C1, C2, RC1 y RC2 cada una con su respectivo número de instancias, número de clientes, depósito central, tamaño de flota y capacidad de cada vehículo.

Tabla 3.1: Instancias de Solomon Para el VRPTW (Conjuntos de Prueba)

Conjunto de Prueba	**No. de Instancias**	**Capacidad de los Vehículos**	**No. de Clientes**	**No. de Vehículos Flota Homogenea**	**No. de Depositos Centrales**
R1	12	200	100	25	1
R2	11	1.000			
C1	9	200			
C2	8	700			
RC1	8	200			
RC2	8	1.000			
TOTAL	**56**				

En la siguiente tabla se observan las diferencias y similitudes existentes entre las 6 categorías en las que se divide el Banco de Prueba propuesto por el profesor Solomon.

Tabla 3.2: Resumen Instancias de Solomon Para el VRPTW

	TW ↓	**TW ↑**	**Los Problemas (Instancias), Comparten las Mismas**
Aleatorio (Random)	R1	R2	Coordenadas Geográficas $(Xcoord, Ycoord)$, Demanda (d_i), Tiempo de Servicio (s_i) = Cte. = 10
Agrupado (Cluster)	C1	C2	Demanda (d_i), Tiempo de Servicio (s_i) = Cte. = 90
Mixto (Random y Cluster)	RC1	RC2	Coordenadas Geográficas $(Xcoord, Ycoord)$, Demanda (d_i), Tiempo de Servicio (s_i) = Cte. = 10
	Capacidad Pequeña	Capacidad Grande	

Donde:

TW: Time Window (Ventana de Tiempo).

TW ↓: Ventanas de Tiempo Estrechas.

TW ↑: Ventanas de Tiempo Amplias.

Cte.: Constante.

Hasta Marzo 24 del 2005, las mejores soluciones conocidas, encontradas por diferentes autores para algunas instancias de prueba, fueron publicadas en la página del profesor Solomon, Universidad del Noreste (Northeastern University, Boston, Massachusetts). Ahí se encuentran las soluciones óptimas de algunas de esas instancias así como las mejores soluciones identificadas con heurísticas.

Pero como luce gráficamente una solución para este tipo de Problemas?

Para responder a esta pregunta primero que todo tomamos como ejemplo una de las instancias de Solomon, en este caso la C101, compuesta, como todas las demás, por un depósito central, representado por el nodo 0 y $n+1$, donde $n = 100$ clientes. Los clientes se encuentran representados por los nodos $\{1, ..,100\}$. Tanto para el depósito central como para cada uno de los clientes, la ubicación geográfica ($XCOORD,\ YCOORD$), demanda ($d_i = DEMAND$), ventanas de tiempo $[a_i, b_i]$ y tiempo de servicio ($s_i = SERVICE\ \ TIME$) son dados, además del tamaño de la flota inicial ($K = VEHICLE\ \ NUMBER$) y la capacidad de cada uno de los vehículos ($C = CAPACITY$).

Donde:

$a_i\ =\ Límite\ Inferior\ de\ la\ Ventana\ de\ Tiempo\ =\ READY\ \ TIME,$

$b_i\ =\ Límite\ Superior\ de\ la\ Ventana\ de\ Tiempo\ =\ DUE\ \ DATE.$

C101

VEHICLE
NUMBER CAPACITY
25 200

CUSTOMER
CUST NO. XCOORD. YCOORD. DEMAND READY TIME DUE DATE SERVICE TIME

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	914	965	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	144	90
25	25	52	40	169	224	90
26	25	55	10	622	701	90
27	23	52	10	261	316	90
28	23	55	20	546	593	90
29	20	50	10	358	405	90
30	20	55	10	449	504	90
31	10	35	20	200	237	90
32	10	40	30	31	100	90
33	8	40	40	87	158	90
34	8	45	20	751	816	90
35	5	35	10	283	344	90
36	5	45	10	665	716	90
37	2	40	20	383	434	90
38	0	40	30	479	522	90
39	0	45	20	567	624	90
40	35	30	10	264	321	90
41	35	32	10	166	235	90
42	33	32	20	68	149	90
43	33	35	10	16	80	90
44	32	30	10	359	412	90
45	30	30	10	541	600	90
46	30	32	30	448	509	90
47	30	35	10	1054	1127	90
48	28	30	10	632	693	90
49	28	35	10	1001	1066	90
50	26	32	10	815	880	90
51	25	30	10	725	786	90
52	25	35	10	912	969	90
53	44	5	20	286	347	90
54	42	10	40	186	257	90
55	42	15	10	95	158	90
56	40	5	30	385	436	90
57	40	15	40	35	87	90
58	38	5	30	471	534	90
59	38	15	10	651	740	90
60	35	5	20	562	629	90
61	50	30	10	531	610	90
62	50	35	20	262	317	90
63	50	40	50	171	218	90
64	48	30	10	632	693	90
65	48	40	10	76	129	90
66	47	35	10	826	875	90
67	47	40	10	12	77	90
68	45	30	10	734	777	90
69	45	35	10	916	969	90
70	95	30	30	387	456	90
71	95	35	20	293	360	90
72	53	30	10	450	505	90
73	92	30	10	478	551	90
74	53	35	50	353	412	90
75	45	65	20	997	1068	90
76	90	35	10	203	260	90
77	88	30	10	574	643	90
78	88	35	20	109	170	90
79	87	30	10	668	731	90
80	85	25	10	769	820	90
81	85	35	30	47	124	90
82	75	55	20	369	420	90
83	72	55	10	265	338	90
84	70	58	20	458	523	90
85	68	60	30	555	612	90
86	66	55	10	173	238	90
87	65	55	20	85	144	90
88	65	60	30	645	708	90
89	63	58	10	737	802	90
90	60	55	10	20	84	90
91	60	60	10	836	889	90
92	67	85	20	368	441	90
93	65	85	40	475	518	90
94	65	82	10	285	336	90
95	62	80	30	196	239	90
96	60	80	10	95	156	90
97	60	85	30	561	622	90
98	58	75	20	30	84	90
99	55	80	10	743	820	90
100	55	85	20	647	726	90

Figura 3.5: Instancia de Solomon C101.100

En la **Figura** C**.5** se observa como lucen los archivos .txt en los cuales se encuentran registrados los datos de cada una de las instancias de Solomon, para este caso se muestran los datos correspondientes a la instancia C101.100.

En la **Figura** C**.6** se puede apreciar, a manera de ejemplo, como luce una solución óptima para una de las instancias de Solomon, en este caso se graficó la solución óptima alcanzada por (KDMSS, 1999) empleando métodos exactos para el problema C101 con 100 clientes. En ella se observa, que los clientes del problema Tipo C1 se encuentran geográficamente divididos en clusters. Cada color representa una ruta asignada a uno y sólo un vehículo, también encontramos que debido a que este tipo de problemas tiene unas ventanas de tiempo muy restrictivas los vehículos pueden atender a pocos clientes en cada ruta (entre 8 a 13 Clientes).

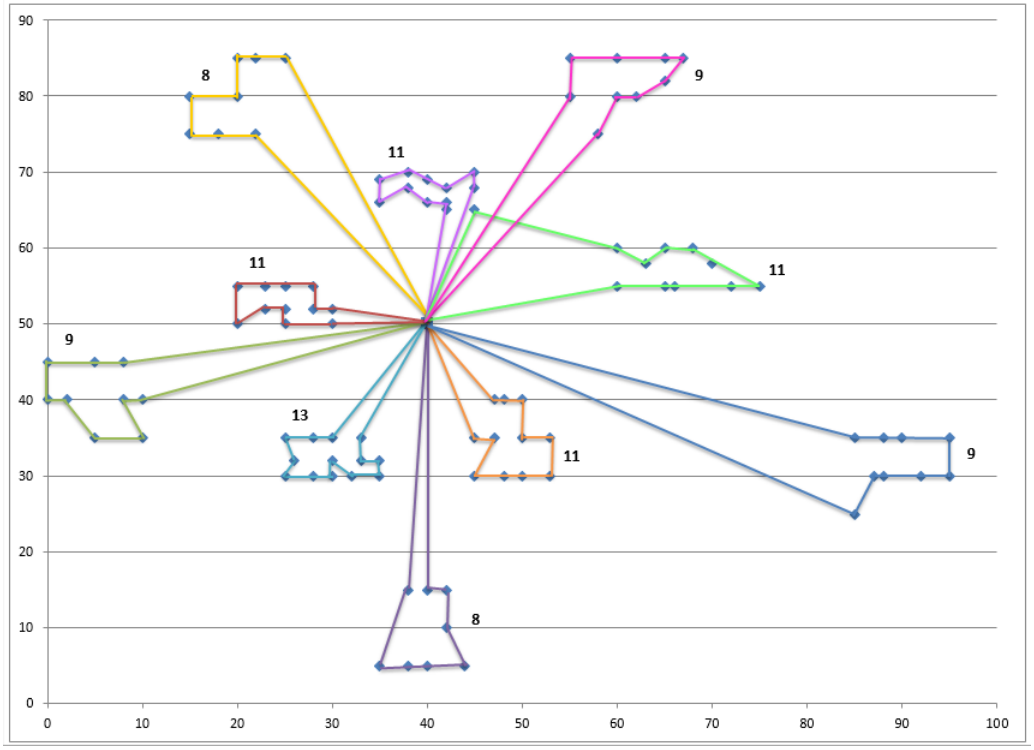

Figura 3.6: Solución Óptima para el Problema C101.100

Entonces, la Solución Óptima es:

DISTANCIA = 827.3

NÚMERO DE VEHÍCULOS = 10 (10 RUTAS):

Tabla 3.3: Solución Óptima para el Problema C101.100

R1: 8 Clientes	**R6:** 11 Clientes
R2: 11 Clientes	**R7:** 9 Clientes
R3: 9 Clientes	**R8:** 11 Clientes
R4: 13 Clientes	**R9:** 9 Clientes
R5: 8 Clientes	**R10:** 11 Clientes
TOTAL	100 Clientes 10 Vehículos Distancia Total = 827.3

Esto quiere decir que para servir de manera óptima a todos los 100 clientes que componen la instancia C101 de Solomon, respetando tanto las restricciones de capacidad de los vehículos como las ventanas de tiempo de cada uno de los clientes y del depósito central, se necesita una flota compuesta por 10 vehículos, los cuales deberán recorrer la distancia total de 827.3

Instancias de Gehring y Homberger para el VRPTW (Extended Solomon´s VRPTW Instances): **?** extendieron los conjuntos de prueba de Solomon, presentando nuevas instancias con 200, 400, 600, 800 y 1000 clientes. A pesar del número de clientes, las nuevas instancias se diseñaron, en la medida de lo posible, de la misma manera que los conjuntos de prueba originales de 100 clientes. Esto concierne especialmente al tamaño de la flota, capacidad del vehículo, tiempo de viaje del vehículo, distribución espacial de los clientes y la densidad y ancho de la ventana de tiempo. Como las originales, las nuevas instancias se dividen en tres categorías: tipo C (clusters de clientes), tipo R (clientes uniformemente distribuidos) y tipo RC (una mezcla de tipo R y C). Dos conjuntos de prueba son propuestos para cada una de estas tres categorías.

Tabla 3.4: Instancias de Gehring y Homberger Para el VRPTW (Conjuntos de Prueba)

No. de Instancias/ Conjunto de Prueba	10						
Conjunto de Prueba	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**	
Capacidad de los Vehículos	**200**	**1.000**	**200**	**700**	**200**	**1.000**	
200 Clientes	50						**No. de Vehículos**
400 Clientes	100						
600 Clientes	150						
800 Clientes	200						
1.000 Clientes	250						

En la página de Homberger, Universidad de Hagen (FernUniversitat in Hagen), piensan publicar, en un futuro cercano, un ranking con las mejores soluciones obtenidas para las instancias extendidas de Solomon (Instancias de Gehring y Homberger). Pero, hasta la fecha, no han publicado resultados. Sin embargo, en el Portal de Transporte y Optimización (TOP Transportation Optimization Portal) que hace parte del grupo SINTEF (la organización de investigación independiente más grande en Escandinavia) ya se han publicado las mejores soluciones conocidas para las instancias de 200, 400, 600, 800 y 1.000 clientes. También se publicaron unos cuantos resultados nuevos par las Instancias de Solomon de 100 clientes.

3.2.2. Marco Teórico

El presente capítulo, presenta las definiciones necesarias que fundamentan teóricamente en forma básica y sintetizada el presente trabajo.

3.2.2.1. API

Una API ("Application Programming Interface") son las reglas y especificaciones que las aplicaciones siguen para comunicarse entre ellas: sirviendo de interfaz entre programas dife-

rentes. Las API son valiosas, ante todo, porque permiten hacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas), reutilizando así código que se sabe que está probado y que funciona correctamente.

3.2.2.2. Optimización:

Según **?**, en el campo de la ingeniería no basta con resolver un problema, se debe encontrar la mejor solución posible, es decir, la solución óptima del problema. Se entiende por solución óptima aquella que cumple con las restricciones del problema y además representa la mejor solución de todas las posibles soluciones factibles. Las soluciones factibles son todas aquellas soluciones que cumplen con las restricciones propuestas por el problema. Consecuentemente, la factibilidad excluye a todas las soluciones que no cumplen con las restricciones del problema, a este conjunto de soluciones se les denomina soluciones no factibles.

Optimización Combinatoria La optimización combinatoria es un campo de las matemáticas aplicadas, que combina técnicas de combinatoria, programación lineal y la teoría de algoritmos para resolver problemas sobre estructuras discretas (**?**).

En esta área se estudian técnicas que permitan resolver problemas cuya naturaleza son NP, y no es posible encontrar una solución de manera exhaustiva.

Según **?** Los problemas de optimización combinatoria contienen los dos elementos que hacen atractivo un problema a los matemáticos: planteamiento sencillo y dificultad de resolución.

De acuerdo a **?**, los problemas combinatorios pueden ser clasificados en dos grandes grupos considerando la existencia o no de algoritmos polinomiales para resolver cada tipo de problema. El primero es el problema en P (Tiempo Polinomial) para el cual existen algoritmos con esfuerzos computacionales de tipo polinomial para encontrar la solución óptima;

y el segundo es el tipo NP (Tiempo Polinomial No Determinista), para el cual no se conocen algoritmos con esfuerzos computacionales de tipo polinomial para encontrar la solución óptima.

En general, los problemas de optimización combinatoria presentan cuatro tipos de variantes: según si los datos del problema son exactos o no, determinísticas o estocásticas; y según si estos se conocen totalmente al momento de resolverlo, estáticas o dinámicas.

Optimización Mono-Objetivo: Según **?**, los problemas de optimización son comúnmente representados en un esquema mono-objetivo. En otras palabras, el proceso consiste en optimizar una sola función objetivo teniendo en cuenta una serie de restricciones que están basadas en restricciones del mundo real. Un problema de optimización mono-objetivo es expresado de la siguiente manera:

Optimizar $[Minimizar/Maximizar]f(x)$ Sujeta a:
$H(x) = 0$
$G(x) \leq 0$

En este caso, la función a optimizar (maximizar ó minimizar) es $f(x)$, donde el vector x es el conjunto de variables independientes. Las funciones $H(x)$ y $G(x)$ son las restricciones del modelo.

Para este problema existen tres conjuntos soluciones: el conjunto universal, el cual está compuesto por todos los posibles valores de x sean factibles ó no. El conjunto de soluciones factibles, el cual está compuesto por todos los valores de x que cumplen con las restricciones$H(x)$ y $G(x)$. Por último, el conjunto de soluciones óptimas, el cual está compuesto por todos los valores de x que, en adición de ser factibles, alcanza el valor óptimo

(mínimo ó máximo) de la función $f(x)$, sea en un intervalo específico $[a, b]$ ó en un contexto global $(-\infty, \infty)$.

Optimización Multiobjetivo: Sea $x^{\longrightarrow} = [x_1, x_2, ..., x_n]^T$ el vector de las variables de decisión, $g_i(x^{\longrightarrow}) \geq 0, i = 1, ..., m$, y $h_i(x^{\longrightarrow}) = 0, i = 1, ..., p$, son las restricciones de desigualdad e igualdad y $f(x^{\longrightarrow}) = [f_1(x^{\longrightarrow}), f_2(x^{\longrightarrow}), ..., f_k(x^{\longrightarrow})]^T$ es el vector de las funciones objetivo que se van a minimizar. Por lo tanto los problemas de la forma:

$f^{\longrightarrow}(x^{\longrightarrow}) \longrightarrow$ "min",

$f^{\longrightarrow} : \Omega \longrightarrow \mathbb{R}^k$,

$\Omega = \{x^{\longrightarrow} \in \mathbb{R}^n : g^{\longrightarrow}(x^{\longrightarrow}) \geq 0, h^{\longrightarrow}(x^{\longrightarrow}) = 0\}$

Se llaman problemas de optimización multiobjetivo. En esta formulación la notación en la primera linea indica la optimización de la función vectorial. Sin embargo, el significado del termino " optimo ", cambia en el caso de la optimización multiobjetivo porqué hay más de una función objetivo que usualmente entran en conflicto con los demás.

Esto significa que una disminución en algunos criterios conduce a un aumento simultáneo de al menos otro criterio. El objetivo es encontrar una solución factible $x^{*\longrightarrow}$ que representa el mejor compromiso entre las funciones objetivas individuales. El óptimo en este caso es llamado Pareto óptimo.

Óptimo de Pareto: Un punto $x^{*\longrightarrow} \in \Omega$ es un Óptimo de Pareto si para cada $x^{\longrightarrow} \in \Omega$ e $I = \{1, 2, ..., k\}$ o bien, $f_i(x^{\longrightarrow}) = f_i(x^{*\longrightarrow}), \forall i \in I$, o, por lo menos hay un $i \in I$, de tal modo que $f_i(x^{\longrightarrow}) > f_i(x^{*\longrightarrow})$

En otras palabras, no hay $x^{\longrightarrow}$ factible que sea al menos tan bueno como $x^{*\longrightarrow}$ para todos los criterios y estrictamente mejor para al menos uno.

Estas soluciones Óptimas de Pareto se consideran las soluciones óptimas en el sentido de los problemas de optimización multi-objetivo.

Dominio de Pareto: Un vector $u^{\longrightarrow} = (u_1, ..., u_k)$ se dice que domina a $v^{\longrightarrow} = (v_1, ..., v_k)$ si y sólo si $\forall i \in \{1, ..., k\} : u_i \leq v_i \bigwedge \exists i \in \{1, ..., k\}$ con $u_i < b_i$. el dominio de Pareto de $u^{\longrightarrow}$ sobre $v^{\longrightarrow}$ se denota por $u^{\longrightarrow} \preceq v^{\longrightarrow}$.

Conjunto de Pareto Óptimo: Para el problema de la forma:
$f^{\longrightarrow}(x^{\longrightarrow}) \longrightarrow$ "min",

$f^{\longrightarrow} : \Omega \longrightarrow \mathbb{R}^k$,

$\Omega = \{x^{\longrightarrow} \in \mathbb{R}^n : g^{\longrightarrow}(x^{\longrightarrow}) \geq 0, h^{\longrightarrow}(x^{\longrightarrow}) = 0\}$

el conjunto Óptimo de Pareto (P^*) se define como $P^* := \{x \in \Omega : \neg\exists x' \in \Omega \, con \, f^{\longrightarrow}(x') \preceq f^{\longrightarrow}(x)\}$

Frente de Pareto: Para el problema de la forma:
$f^{\longrightarrow}(x^{\longrightarrow}) \longrightarrow$ "min",

$f^{\longrightarrow} : \Omega \longrightarrow \mathbb{R}^k$,

$\Omega = \{x^{\longrightarrow} \in \mathbb{R}^n : g^{\longrightarrow}(x^{\longrightarrow}) \geq 0, h^{\longrightarrow}(x^{\longrightarrow}) = 0\}$

y el conjunto Óptimo de Pareto P^*, el Frente de Pareto (FP^*) se define como $FP^* := \{u^{\longrightarrow} = f^{\longrightarrow} = (f_1(x), ..., f_k(x)), x \in P^*\}$

Soluciones Óptimas de Pareto también se denominan no inferiores, admisibles o solucione eficientes. Sus vectores correspondientes $f^{\longrightarrow}(x^{*\longrightarrow}) = [f_1(x^{*\longrightarrow}), ..., f_k(x^{*\longrightarrow})]^T$ se denominan no dominadas. El Conjunto Óptimo de Pareto consiste en las soluciones cuyos corres-

pondientes vectores son no dominados respecto a todos los otros vectores de comparación. Si los vectores no dominados son graficados en el espacio objetivo, son conocidos como Frente de Pareto.

Las técnicas de solución para problemas de optimización multiobjetivo consiste generalmente de dos etapas: La optimización de las múltiples funciones objetivo, y la decisión que tipo de "trade-offs" son más apropiados desde el punto de vista del tomador de decisiones.

3.2.2.3. Problemas de Enrutamiento

Los problemas de enrutamiento han sido ampliamente estudiados debido al sinnúmero de aplicaciones reales e importancia económica, estos involucran la generación de un tour, o una colección de tours, en una red, o a un subconjunto de una red, dado un conjunto de restricciones y la necesidad de optimizar uno o varios objetivos fijados. Dichos problemas son usualmente configurados con un sólo objetivo, generalmente el de minimizar costos, aun cuando la mayoría de problemas que se encuentran en la industria, particularmente en logística, son multiobjetivos por naturaleza. En la vida real, por ejemplo, puede haber muchos costos asociados con un solo tour. Por otra parte, los objetivos puede que no siempre estén limitados a los costos. De hecho, otros aspectos, tales como balancear cargas de trabajo (tiempo, distancia, entre otras), pueden ser tomados en cuenta simplemente adicionando nuevos objetivos.

Según **?**, un problema de enrutamiento puede definirse en términos de los siguientes componentes: la Red, la Demanda(s), la Flota, el Costo(s), y el Objetivo u Objetivos:

- **La Red:** puede ser simétrica, asimétrica o mixta. Es representada por un grafo en el cual los nodos simbolizan las ciudades, los clientes y/o las bodegas, y los arcos representan vínculos reales, por ejemplo: caminos, tuberías, o conexiones simbólicas.

En algunos problemas se pueden asociar ventanas de tiempo a los nodos (como en el VRPTW) o a los arcos.

- **Las Demandas:** pueden ser fijas o estocásticas, se puede asociar tanto a los nodos como a los arcos, y se puede dar para uno o varios productos.

- **La Flota:** Una flota puede ser heterogénea u homogénea. Puede estar compuesta por un solo vehículo o varios vehículos, cuyo uso, puede o no, por ejemplo, estar limitada por la capacidad, tiempo o distancia. Además, pueden existir dependencias entre los vehículos, conductores, productos, nodos y/o arcos. El término flota no siempre se refiere a un grupo de vehículos, de hecho, en algunos problemas no hay vehículos.

- **Los Costos:** generalmente son fijos para el vehículo y variables para su uso, en términos de distancia recorrida o tiempo empleado. Los costos también pueden incluir las penalizaciones al servicio en las que se incurre cuando un cliente recibe tarde o incompleto un pedido. Relacionado con los costos, los beneficios también pueden ser asociados a los nodos/arcos, y el beneficio se consigue cuando el nodo es visitado o el arco es seleccionado.

- **Los Objetivos:** el problema puede tener un único objetivo (mono-objetivo), o puede tener múltiples y diversos objetivos (multiobjetivo). La función objetivo puede ser computada para un solo período o para varios períodos, aunque en éste último caso, los vehículos y visitas deben ser asignadas a los diferentes períodos. Los objetivos más comunes incluyen minimizar el total de la distancia recorrida, el total del tiempo requerido, el costo total del tour, y/o el tamaño de la flota, y maximizar la calidad del servicio y/o el beneficio conseguido. Sin embargo, cuando múltiples objetivos son identificados, los distintos objetivos frecuentemente entran en conflicto. Por esta razón adoptar un punto de vista multiobjetivo puede ser ventajoso.

  En cuanto a los objetivos, **?**, los clasifican de acuerdo a los componentes del problema con los que se encuentran relacionados: Tour, Nodo/Arco ó Recursos:

- **Entre los objetivos relacionados con el Tour se encuentran:** Costo, Makespan, Balance, Objetivos Específicos. Minimizar el costo de las soluciones generadas es el objetivo más común. Los Costos pueden ser expresados como: la distancia recorrida, el tiempo requerido, o el número de clientes visitados. El Makespan, es el objetivo menos utilizado, este busca minimizar la distancia del tour más largo. Respecto al Balanceo, son objetivos que frecuentemente se introducen para incluir un elemento de igualdad equilibrando disparidades entre los tours. En cuanto a los Objetivos Específicos, estos son definidos por la naturaleza del problema o en el contexto en el que se desarrolla el mismo.

- **Entre los objetivos relacionados con los Nodos/Arcos:** la ventana de tiempo es reemplazada por un objetivo que minimiza el número de restricciones violadas, el tiempo total de espera del conductor o del cliente debido a la llegada temprano o tarde del vehículo, o ambos objetivos al mismo tiempo.

- **Entre los objetivos relacionados con los Recursos:** un objetivo que frecuentemente aparece es la minimización del número de vehículos.

Según **?**, una detallada, mas no exhaustiva, lista de subcategorías para el problema generalizado de enrutamiento es la siguiente: 1) el Problema de Ruta Mínima (shortest path problem), 2) El Problema del Cartero Chino (Chinese postman problem), 3) el Problema del Cartero Rural (rural postman problem), 4) Dial-a-Ride Service Route Problem, 5) El Problema de Rutas por Arcos (Arc Routing Problem), 6) TSP, y 7) VRP. Esta clasificación claramente delimita y aísla al VRP de los demás tipos de problemas de enrutamiento, los cuales, se diferencian por el escenario de enrutamiento, restricciones implicadas, y por ende los requerimientos del moldeamiento matemático impuesto. El VRP ha generado una literatura bastante extensa para permitirle ser considerado como un campo de conocimiento separado y distinto. El creciente interés en el VRP hace que la elaboración sistemática de este campo sea crucial para ayudar a los actuales investigadores así como para atraer nuevos

potenciales al campo. A continuación se brinda una definición para el TSP y el VRP por su importancia académica y múltiples aplicaciones en la vida real:

El Problema del Agente Viajero TSP (Travelling Salesman Problem): De acuerdo a **?**, el problema del agente viajero, constituye la situación general y de partida para formular otros problemas combinatorios más complejos, aunque más prácticos, como el enrutamiento de vehículos VRP y la programación de tareas dependientes del tiempo de alistamiento. El TSP estándar es un problema NP-Hard que consiste en un agente vendedor cuya misión es visitar un conjunto de ciudades o clientes. En una jornada el agente viajero parte desde su oficina y debe visitar todos sus clientes, máximo una vez, y finalmente debe regresar a su punto de origen al final de la jornada. El agente debe tomar la ruta que minimice los costos, en general definidos como función de la distancia recorrida o el tiempo trascurrido, aunque otras definiciones aplican también a la caracterización del problema.

El Problema de Enrutamiento de Vehiculos VRP (Vehicle Routing Problem): Es una extensión del problema del TSP en el cual no sólo se visitan todos los clientes, sino que se satisfacen sus demandas. En el VRP, una flota de vehículos con capacidad restringida parte desde un centro de distribución (o bodega) para abastecer a un conjunto de clientes al mínimo costo. La capacidad de la flota no puede ser excedida, luego el problema se concentra en encontrar una ruta para cada vehículo tal que la sumatoria de los costos de los trayectos recorridos sea minimizada.

? desarrollaron la siguiente revisión taxonomíca del VRP, que surge a raíz de la amplitud y disparidad de la literatura que trata a este problema combinatorial que trasciende varias disciplinas y profesiones, lo que dificulta el seguimiento de su desarrollo. Tal marco propuesto permite la clasificación de todos los artículos sobre VRP y, alternadamente, la

identificación sistemática de vacíos en la literatura guiando así hacia los asuntos potenciales de investigación.

Esta fue construida de forma arborescente. Los niveles de ramificación de arriba hacia abajo son, a lo más, tres con el fin de proveer coherencia y no sacrificar nada en términos de comprensión. El primer nivel de ramificación permite clasificar a los artículos sobre el VRP por: (1) Tipo de Estudio (type of study), (2) Características del Escenario (scenario characteristics), (3) Características Físicas del Problema (problem physical characteristics), (4) Características de la Información (information characteristics), y (5) Características de los Datos (data characteristics). Dentro del primer conjunto de características, Tipo de Estudio, un artículo es diferenciado basado en su contenido y si encaja en una de las cuatro subcategorías presentadas. Después de identificado el tipo de estudio, para aquellos artículos que comprendan un problema, se introducen el resto de categorías. En la segunda categoría, Características del Escenario, aquellos factores que no son parte de las restricciones integradas en la solución, pero parte del escenario del problema, son listadas. La tercera categoría, Categorías Físicas del Problema, comprende aquellos factores que directamente afectan la solución. La cuarta categoría, Características de la Información, tiene que ver con la solución del VRP presentada clasificando la naturaleza, accesibilidad, y procesamiento de la información. Aunque la última categoría, Características de los Datos, posee un nombre similar al de la categoría previa, introduce un propósito distintivo: clasificar el tipo de datos basado en su origen.

Empleando un grupo de artículos que representan métodos ligeramente diferentes y que tratan diferentes tópicos del VRP. Ellos encontraron que la taxonomía desarrollada durante su investigación, ha demostrado ser lo suficientemente robusta para incluir un conjunto diverso de artículos sobre el VRP, al igual que todas las publicaciones de taxonomías previas en este campo.

VRP con Ventanas de Tiempo (VRPTW): Este problema es una extensión del bien conocido VRP (**?**); (**?**) al que **?** adicionó restricciones de ventanas de tiempo.

Según **?**, el VRPTW, consiste en diseñar un conjunto de rutas a un mínimo costo, que se originan y terminan en un depósito central, para una flota de vehículos que debe servir a una serie de clientes con demandas conocidas dentro de un intervalo de tiempo previamente establecido, el cual es llamado ventana de tiempo. Los clientes deben ser asignados exactamente una vez a los vehículos tal que las capacidades de estos no sean excedidas.

Las ventanas de tiempo pueden ser duras (VRPHTW) o suaves (VRPSTW). En el caso de las ventanas de tiempo duras, si un vehículo llega demasiado temprano a donde un cliente (límite inferior ventana de tiempo), se le es permitido esperar hasta que el cliente se encuentre listo para empezar el servicio. Sin embargo, a un vehículo no le es permitido llegar a un nodo después del tiempo más tarde para prestar el servicio (límite superior de la ventana de tiempo). En contraste, en el caso de las ventanas de tiempo suaves, se permite que estas sean violadas pero asociadas con una penalización. Cabe resaltar que la mayoría de los esfuerzos de investigación se han dirigido a la variante de las ventanas de tiempo duras.

Las ventanas de tiempo se presentan naturalmente en problemas que deben enfrentar a diario las organizaciones de negocios. Ejemplos específicos de problemas con ventanas de tiempo dura son: entregas del banco, entregas postales, entre otros. Entre los problemas con ventanas de tiempo suaves, los problemas de pedidos constituyen un ejemplo importante.

Objetivo: Minimizar el número de vehículos y la suma del tiempo de viaje y espera necesarios para proveer a todos los clientes en sus horas requeridas.

Factibilidad: El VRPTW es caracterizado por las siguientes restricciones adicionales:

- Una solución llega a ser infactible si proveen carga a un cliente después del límite superior de su ventana de tiempo.

- La llegada de un vehículo antes del límite más bajo de la ventana de tiempo causa tiempo de espera adicional en la ruta.
- Cada ruta debe iniciar y finalizar dentro de la ventana de tiempo asociada al depósito central.
- En el caso de ventanas de tiempo suaves, un servicio posterior no afecta la factibilidad de la solución, pero es penalizada agregando un valor a la función objetivo.

El VRPTW tiene múltiples objetivos que pretenden minimizar no sólo el número de vehículos requeridos sino también el tiempo de viaje, tiempo de espera y distancia total recorrida por la flota de vehículos. Para el caso de las Ventanas Suavizadas, también se busca minimizar las penalizaciones obtenidas al violar las ventanas de tiempo.

3.3. OBJETIVOS

3.3.1. Objetivo General

Desarrollar una herramienta, para resolver el Problema Combinatorio Multiobjetivo de Enrutamiento de Vehículos con Ventanas de Tiempo Duras (VRPHTW), empleando la API de Google Maps.

3.3.2. Objetivo Específicos

- Desarrollar un modelo informático−matemático para resolver el VRPHTW considerando la congestión vehicular.
- Determinar el método de medición de la congestión vehicular.
- Desarrollar una herramienta informática, para resolver el modelo propuesto sometido a rutas con congestión en determinadas horas del día.
- Verificar la herramienta informática desarrollada, empleando instancias de prueba construidas a partir de ubicaciones reales.

3.4. METODOLOGÍA

- Etapa 1. Consiste en definir el área de estudio en la que se enfocará el trabajo de tesis. Se procederá a realizar una revisión exhaustiva del estado del arte en el problema de enrutamiento de vehículos, se buscará innovar en el enfoque de solución del problema, basándose en el estado del arte encontrado.

  Durante esta etapa se realizó una revisión literaria exhaustiva, consultando Bases de Datos especializadas: Science Direct (Consorcio Colciencias-Elsevier), SCOPUS (Consorcio Colciencias-Elsevier), PROQUEST, JSTOR y EBSCO. En busca de información respecto a investigaciones previas relacionadas con el ruteo de vehículos, específicamente con el VRPTW y sus variantes, con el fin de determinar y clasificar las metodologías anteriormente desarrolladas por otros autores, especialmente las que han sido más ampliamente empleadas gracias a sus buenos resultados y tratar de determinar hacia donde se dirigen estas investigaciones en el futuro próximo, al igual que la pertinencia del presente proyecto.

  Los criterios de búsqueda empleados fueron: VRP y VRPTW, y en cuanto al tipo de documento consultado, en esta primera etapa de la investigación se revisaron publicaciones en: libros, revistas y algunas tesis de grado relacionadas con el problema objeto de estudio.

- Etapa 2. Concluir con la definición del área de estudio en la que se enfocará la investigación. Validar el diseño del modelo matemático que considere las condiciones del problema lo más cercano a la realidad posible, y la descripción de los supuestos que se consideren. El modelo se estudiará para conocer en profundidad su comportamiento bajo diferentes escenarios.

  Con los hallazgos de la Primera Etapa del Proyecto, en esta Segunda Estapa, se procedió a seleccionar el modelo de enrutamiento de vehículos a resolver:

Este proyecto se enfoca en resolver un problema combinatorio de optimización multiobjetivo de enrutamiento de vehículos con ventanas de tiempo duras, el VRPHTW, el cual es una de las variantes del VRPTW, debido a su importancia académica y económica, ya que se puede utilizar para modelar muchos problemas del mundo real.

Durante esta etapa se realizó una segunda revisión literaria exhaustiva, consultando Bases de Datos especializadas: Science Direct (Consorcio Colciencias-Elsevier), SCOPUS (Consorcio Colciencias-Elsevier), PROQUEST, JSTOR y EBSCO. Pero esta vez en busca de información respecto a investigaciones previas relacionadas con el VRPHTW, para determinar y clasificar las metodologías existentes, especialmente las que arrojan mejores resultados y determinar hacia donde se dirigen estas investigaciones en el futuro próximo, con el fin de innovar en el enfoque de solución del problema, basándose en el estado del arte encontrado, al igual que para reafirmar la pertinencia del presente proyecto.

Los criterios de búsqueda empleados fueron: VRPHTW, y en cuanto al tipo de documento consultado, en esta segunda etapa de la investigación se enfocó sólo en artículos científicos encontrados.

Durante esta estapa se seleccionó el modelo matemático del Problema de Enrutamiento de Vehículos con Ventanas de Tiempo Duras (VRPHTW),sometido a rutas con congestión en determinadas horas del día, con dos funciones objetivo: costo total de transporte y tiempo total de espera , evitando las rutas congestionadas, se espera obtener reducción en los costos totales y tiempos de espera.

- Etapa 3. Establecer bajo qué condiciones existe la necesidad de usar métodos no exactos para solucionar el modelo planteado y diseñar una metodología de solución alternativa. Comprobar su destreza y experimentar con valores de los parámetros para obtener la mejor solución posible. Así mismo, se evaluará el comportamiento de nuestra metodología de solución en diferentes escenarios.

Durante esta etapa, y después de clasificar las metodologías existentes en la literatura del VRPHTW, y teniendo en cuenta la duración de la investigación, se optó por acudir al uso de las API que sirven de interfaz entre programas diferentes. En vista del gran potencial que esconde la plataforma online: Google Maps, y a su gran popularidad, se decidió aprovechar esta, con el fin de resolver problemas de enrutamiento de vehículos cada vez más cercanos a la realidad. A través de: RuGle, un script desarrollado en el lenguaje de programación de código abierto: Ruby, y basado en la API de Google, con el fin de encontrar la solución al VRPHTW multiobjetivo, sometido a rutas con congestión en determinadas horas del día.

- Etapa 4. Aplicar la metodología propuesta al caso de estudio seleccionado.

  Durante esta etapa se empleó la herramienta desarrollada para resolver el problema de optimización combinatoria multiobjetivo: VRPHTW sometido a rutas con congestión en determinadas horas del día. Se construyeron dos instancias de prueba de 4 y 10 nodos, las cuales se corrieron teniendo en cuenta y no teniendo en cuenta el tráfico, los resultados arrojados se verificaron manualmente para comprobar su factibilidad y se presentarón los respectivos resultados obtenidos al igual que las conclusiones de la investigación.

3.5. RESULTADOS \ PRODUCTOS ESPERADOS Y POTENCIALES BENEFICIARIOS

En esta sección, se presentan los productos y resultados esperados, al igual que los potenciales beneficiarios de este proyecto. Así como los relacionados con la generación de conocimiento y \ o nuevos desarrollos tecnológicos, y los conducentes al fortalecimiento de la capacidad científica nacional.

3.5.1. Relacionados con la Generación de Conocimiento y \ o Nuevos Desarrollos Tecnológicos

Resultados de generación de nuevo conocimiento: Un artículo de investigación que contenga la explicación de la estrategia de solución desarrollada para resolver el modelo de Enrutamiento de Vehículos con Ventanas de Tiempo Duras VRPHTW multiobjetivo, sometido a rutas con congestión, se espera que el artículo sea publicable en una revista indexada internacional.

Una herramienta autodidáctica y de código abierto, para resolver el VRPHTW multiobjetivo, empleando la API de Google Maps, que permita encontrar soluciones aproximadas.

3.5.2. Conducentes al Fortalecimiento de la Capacidad Científica Nacional

Al igual que el sometimiento de un artículo científico de investigación a una revista indexada internacional, se espera como resultado el desarrollo de una disertación como requisito para obtener el título de Maestría en Ingenierí Industrial.

3.5.3. Dirigidos a la apropiación social del conocimiento

Con RuGle, se pretende potenciar el acceso ágil a la información y al intercambio de conocimiento de un modo sencillo y didáctico con todos aquellos que se inician en el campo de los problemas de enrutamiento de vehículos con ventanas de tiempo, a tráves de una herramienta que les permita encontrar soluciones tabuladas y gráficas a VRPHTW sometidos a rutas con congestiÃşn para cualquier lugar del planeta, gracias a que está construida sobre la plataforma de Google Maps, la cual abarca todo el globo térraqueo y cuenta con mapas digitales actualizados de cualquier zona del mundo. Y cuyo código pueden modificar para generar soluciones alternativas a las propuestas por los autores, permitiéndoles generar nuevo conocimiento a partir de una herramienta de autodidáctica y de fácil manejo.

3.6. IMPACTO ESPERADO

Gracias a los avances tecnológicos de los últimos años, acceder a mapas digitales con un gran nivel de detalle y sofisticación se ha hecho cada vez más sencillo, permitiéndole al usuario una mayor interacción con ellos, debido a que son capaces de llegar a localizar lugares específicos y trazar rutas, además, gracias a la introducción de las herramientas de búsqueda y el movimiento en el mapa, se ha despertado aún más el interés en el uso de las imágenes satelitales, tanto para fines investigativos como para usos personales. Un popular servidor de aplicaciones de mapas en la web es Google Maps, el cual ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calles.

Con la presente investigación, se pretende aprovechar el gran potencial que esconde este tipo de software, a través de de RuGle, una interfaz sencilla e intuitiva, capaz de obtener una solución bastante buena a un problema multiobjetivo de enrutamiento de vehículos con ventanas de tiempo duras (VRPHTW), en el que se tiene que hallar las mejores rutas, teniendo en cuenta la congestión vehicular, entre un conjunto de nodos geográficamente dispersos sometidos a restricciones temporales de servicio. RuGle es una herramienta autodidáctica muy útil para aquellos investigadores que se inician por primera vez en el mundo de los VRPHTW.

Al difundir la investigación en contexto a través de las universidades, centros de investigación y empresas privadas y públicas, debido a su importancia cómo herramienta de solución alternativa, en una variedad de campos de aplicación, como: transporte, logística, comunicaciones, manufactura, operaciones militares, entre otros. Se espera aportar nuevas alternativas para la solución de problemas de enrutamiento de vehículos, no sólo para la distribución y entrega de mercancías, sino para futuras investigaciones tendientes a disminuir los elevados niveles de congestión vehicular, consumo de energía de los vehículos de carga, así como el

impacto medioambiental negativo, tales como: emisiones, accidentes, ruido y vibraciones. Se encontró que en años recientes el problema que más le ha interesado a la logística urbana ha sido el de optimizar globalmente los sistemas de transporte, tanto de pasajeros como de mercancías, considerando los costos y beneficios de esquemas tanto públicos como privados. Estos conceptos son objeto de interés presentes en muchos modelos desarrollados recientemente para predecir y establecer sistemas inteligentes de transporte. Por lo tanto se espera que la alternativa de solución planteada en este proyecto contribuya no sólo a la optimización de entregas de mercancía sino también a optimizar sistemas de transporte urbano en general que contribuyan a una mejor calidad de vida en las ciudades.

CAPÍTULO 4

DESARROLLO DE LA HERRAMIENTA

En este capítulo se describe RuGle, la herramienta desarrollada durante la presente investigación, que incluye: el modelo matemático del VRPHTW multiobjetivo a resolver, las instancias de prueba, la construcción de soluciones iniciales con las modificaciones realizadas a la heurística de inserción I1 y la complejidad computacional del algoritmo desarrollado.

4.1. MODELO MATEMÁTICO DEL VRPHTW

El modelo clásico del VRPTW tiene dos objetivos que son tratados lexicográficamente, es decir que, primero es minimizado el número de vehículos y luego, para esa solución, es minimizada la distancia. Mientras que, en las investigaciones en VRPHTW multiobjetivo, asignan el mismo nivel de prioridad a ambos objetivos, en lugar de considerarlos lexicográficamente. El modelo que se resolverá será el del VRPHTW sometido a rutas con congestión, en este problema se sigue persiguiendo el fin de entregar mercancía diariamente desde un único deposito central a un número de clientes geográficamente dispersos. Los objetivos son minimizar el costo total de transporte y el tiempo total de espera de la flota. Por lo tanto, el número de vehículos empleados, la distancia total recorrida por ellos y el tiempo total de espera de la flota sometida a rutas con congestión en determinadas horas del día, deben

ser lo más pequeña posible. Evitando vías congestionadas, es posible disminuir el tiempo de espera de la flota vehicular, pero la distancia total recorrida podría incrementarse al optar por vías alternas más largas pero menos transitadas.

El VRPHTW puede ser descrito de la siguiente forma: El grafo (V, A) representa una red donde $V = \{0, ..., n+1\}$ es el conjunto de nodos y A es el conjunto de arcos. Los nodos $\{1, ..., n\}$ representan a los clientes. Se asume que sólo hay un depósito central, el cual se encuentra asociado a los nodos 0 y $n+1$. Los arcos $(i, j) \in A$ son las conexiones entre los clientes. Cada arco $(i, j) \in A$ tiene asociado un costo c_{ij}. Los bucles (i, i), arcos que conectan a un nodo consigo mismo, no están permitidos, y por lo tanto los costos para estos arcos se fijan en $c_{ii} = +\infty, \forall i \in V$. El conjunto de clientes que son directamente alcanzables desde el cliente i se denotan por $\Delta^+(i)$, el llamado inicio hacia delante de i. Análogamente, $\Delta^-(i)$ es el inicio hacia atrás del cliente i, el conjunto de clientes desde el cual el cliente i es directamente alcanzable.

La ventana de tiempo del cliente se denota por $[a_i, b_i]$. El conjunto de vehículos idénticos se denota por K. Cada vehículo inicia y finaliza su ruta en el depósito central, cuya ventana de tiempo es denotada por $[a_0, b_0] = [a_{n+1}, b_{n+1}]$ donde el límite inferior representa la salida más temprana posible del depósito central. Cuando un vehículo está atendiendo a un cliente, tiene que quedarse ahí por un periodo de tiempo $s_i > 0, i \in V$ (tiempo de servicio). Cada cliente tiene cierta demanda $d_i, i \in V$. Ya que cada cliente puede ser visitado sólo una vez, su demanda debe ser satisfecha en esa visita. El tiempo de servicio y la demanda del depósito central son $s_0 = s_{n+1} = 0$ y $d_0 = d_{n+1} = 0$, respectivamente.

Se asume que los tiempos de viaje desde el cliente i hasta el cliente j, los tiempos de servicio s_i, los intervalos $[a_i, b_i]$, las demandas d_i y la capacidad C de los vehículos están dados.

En el modelo original del VRPHTW, se asume que los tiempos de viaje cumplen la desigualdad triangular $t_{ik} + t_{kj} \geq t_{ij}$, y que la matriz de tiempo de viaje y la matriz de distancia coinciden. Por lo tanto, distancia y tiempo de viaje, son empleadas indistintamente. Para efectos de esta investigación, eso no será así.

El modelo matemático para el VRPHTW contiene dos tipos de variables de decisión, a saber, variables de flujo y tiempo. Las variables de flujo $x_{ijk}, (i,j) \in A, k \in K >$ son iguales a 1 si el vehículo k utiliza el arco $(i,j) \in A$, y 0 de otro modo. Las variables de tiempo $w_{ik}, i \in V, k \in K$, especifican el inicio del servicio donde el cliente i cuando es atendido por el vehículo k. Si i no es atendido por el vehículo k, w_{ik} no tiene sentido.

Si el vehículo k llega dentro de la ventana de tiempo $[a_j, b_j]$ del cliente j, su tiempo de espera es cero. Pero si llega muy temprano a la locación del cliente j, tiene que esperar hasta a_j para poder iniciar el servicio, por lo tanto, el tiempo de espera, por llegadas anticipadas, para el vehículo k es:

$$te_k = a_j - (w_{ik} + s_i + t_{ij}) \qquad [4.1]$$

Si el vehículo llega muy tarde a la locación del cliente, es decir, después de la ventana superior de tiempo b_j. la solución es infactible.

Por lo tanto, los objetivos que deben ser minimizadas simultáneamente son:

- El costo total conformado por los costos fijos de vehículo y costos por distancia recorrida:

$$f_1(x_{ijk}) = \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \qquad [4.2]$$

- El tiempo total de espera de la flota vehicular:

$$f_2(x_{ijk}) = \sum_{k \in K} \sum_{(i,j) \in A} te_k x_{ijk} \qquad [4.3]$$

El problema se puede indicar matemáticamente como:

$$min f_1(x_{ijk}) \tag{4.4}$$

$$min f_2(x_{ijk}) \tag{4.5}$$

Sujeto a:

$$\sum_{k \in K} \sum_{j \in \Delta^+(i)} x_{ijk} = 1 \quad \forall i \in N, \tag{4.6}$$

$$\sum_{j \in \Delta^+(0)} x_{0jk} = 1 \quad \forall k \in K, \tag{4.7}$$

$$\sum_{i \in \Delta^-(j)} x_{ijk} - \sum_{i \in \Delta^+(j)} x_{jik} = 0 \quad \forall k \in K, j \in N, \tag{4.8}$$

$$\sum_{i \in \Delta^-(n+1)} x_{i,n+1,k} = 1 \quad \forall k \in K, \tag{4.9}$$

$$\sum_{k \in K} \sum_{i \notin S} \sum_{i \in S} x_{ijk} \geq 1 \quad \forall S \subset V, \mid S \mid \geq 2, \tag{4.10}$$

$$\sum_{i \in N} d_i \sum_{j \in \Delta^+(i)} x_{ijk} \leq C \quad \forall k \in K, \tag{4.11}$$

$$x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \leq 0, \quad \forall k \in K, (i,j) \in A \tag{4.12}$$

$$a_i \sum_{j \in \Delta^+(i)} x_{ijk} \leq w_{ik} \leq b_i \sum_{j \in \Delta^+(i)} x_{ijk}; \forall k \in K, i \in N \tag{4.13}$$

$$x_{ijk} \in \{0,1\} \quad \forall k \in K, (i,j) \in A \qquad [4.14]$$

Sea $N = V \setminus \{0, n+1\}$ el conjunto de clientes. La restricción **(4.6)** afirma que cada cliente $i \in N$ es asignado a una ruta de vehículo. El conjunto de restricciones **(4.7)**, **(4.8)** y **(4.9)** son las restricciones de flujo. Ellas condicionan que cada vehículo k tiene que abandonar el nodo 0 una vez, que deja al cliente i, $i \in N$, si y sólo si entra en ese cliente, y que tiene que regresar al nodo $n+1$. Para prevenir subtours, la restricción **(4.10)** es requerida. La restricción **(4.11)** garantiza que ningún vehículo pueda atender más clientes de lo que permite su capacidad. Si la suma de las demandas de los clientes despachados por el vehículo k es inferior a la capacidad, el vehículo carga sólo la suma de las demandas. La restricción **(4.12)** expresa que un vehículo k viajando del cliente i a j, no puede arribar donde el cliente j antes de $w_{ik} + s_i + t_{ij}$. La restricción **(4.13)** se asegura de que el inicio del servicio en el cliente i se lleve a cabo dentro de la ventana de tiempo correspondiente. El conjunto de restricciones de integralidad se encuentra dado por **(4.14)**.

4.2. INSTANCIAS DE PRUEBA CONSTRUIDAS PARA EL VRPTW

Debido a las restricciones de las herramientas que se emplearon para el desarrollo de la presente investigación, las instancias a resolver tuvieron que ser relativamente pequeñas, para evitar realizar muchos cálculos mátematicos que agotarán rápidamente la llave gratuita que proporciona Google, la cual cuenta con un número límitado de peticiones no pagas que se le pueden realizar al servidor de Google Maps, esto es, máximo 100.000 peticiones cada 24 horas. Por ésta misma razón, sólo se construyeron dos instancias de prueba, una de 4 (Ver **Tabla 5.1**) y otra de 10 nodos (Ver **Tabla 5.2**). La distribución geográfica de los clientes es aleatoria. La flota es homogénea, de máximo 25 vehículos, y cada vehículo tienen una

capacidad de 200 unidades. La ubicación del depósito central y los clientes, al igual que sus ventanas de tiempo, demandas y tiempos de servicio se encuentran dados. La estructura de dichas instancias es básicamente la misma que las instancias de Solomon y las de Gehring y Homberger. En el **Apéndice A**, **Sección A.1** se explica en detalle como construir estas instancias de prueba para resolver un VRPHTW empleando RuGle.

Tabla 4.1: Instancias de Prueba Contruidas para RuGle

Conjunto de Prueba	**No. de Clientes**	**Capacidad de los Vehículos**	**No. de Vehículos Flota Homogénea**	**No. de Depósitos Centrales**
n4	4	200	25	1
n10	10			

4.3. CODIFICACIÓN EN RUBY CON GOOGLE API: RuGle

Para construir las soluciones del VRPHTW sometido a rutas con congestión a determinadas horas del día, la presente investigación se basó en la clásica heurística de **?**, conocida como Heurística de Inserción I1. Dicha heurística fue modificada y adaptada para poder interactuar con la Plataforma Online de Google Maps. Una de las modificaciones que se le realizó a éste algoritmo codificado en Ruby, fue dejar de utilizar la distancia euclidiana y el tiempo de viaje entre dos puntos, indistintamente. RuGle en cambio, utiliza ubicaciones reales de cualquier lugar del mundo para la construcción de las instancias de prueba, además, para calcular la distancia, tiempo de viaje y tiempo de viaje con estimación de tráfico entre los nodos que componen la red objeto de estudio, se hizo uso del API de Google Maps. Durante la investigación, se construyeron dos instancias de prueba pequeñas utilizando ubicaciones reales en la ciudad de Barranquilla - Colombia. El nuevo algoritmo en vez de calcular los datos antes mencionados, simplemente hace una petición al API de Google Maps, la cual envía información respecto al origen, destino, modo (para éste caso el modo es conduciendo), hora de salida y el modelo de tráfico (para éste caso el modelo de tráfico

se considera cómo pesimista), dicha petición se hace teniendo en cuenta la hora en la cual el vehículo saldrá del nodo origen para atender a los clientes, obteniendo así un enfoque más realista y mejorando un poco más la precisión a la hora de calcular el tiempo y distancia entre los nodos. Uno de los aportes que esta investigación realiza a todos aquellos que se inician en el campo del VRPHTW multiobjetivo, es explicar de manera sencilla los tiempos que componen este problema: Tiempo de Inicio de Servicio, Tiempo de Servicio, Tiempo de Viaje, Tiempo de Llegada y Tiempo de Espera. Y como calcular cada uno de ellos, al igual que los tres casos que se pueden presentar al momento de resolver este tipo de problemas. Tal como se muestra más adelante en esta sección.

4.3.1. Número Mínimo de Vehículos

Se utilizará la ecuación **(4.15) ?**, con el fin de identificar hasta que punto se puede continuar minimizando o no el número de vehículos necesarios para poder atender la demanda total de todos los clientes que conforman la instancia de prueba:

$$k_{min} = [(\sum_{i \in N} d_i)/C], \qquad [4.15]$$

Donde d_i es la demanda del cliente i y asumiendo que todos los vehículos tienen la misma capacidad C.

4.3.2. Tiempos y Casos Particulares en el VRPHTW

Del modelo matemático expuesto en la Sección 4.1, se infiere que, para el caso del VRPHTW:

El tiempo de llegada del vehículo k al cliente j, después de haber servido al cliente i, está dado por:

$$ta_{jk} = w_{ik} + s_i + t_{ij} \quad [4.16]$$

y el tiempo de inicio de servicio del vehículo k en el cliente j se encuentra dado por:

$$w_{jk} = ta_{jk} + te_k \quad [4.17]$$

Por lo tanto, se pueden presentar los siguientes casos:

1. Si $ta_{jk} < a_j$, entonces: $te_k \neq 0$:

   Si el tiempo de llegada del vehículo k a la locación del cliente j es menor que el tiempo en el que dicho cliente acepta empezar a ser atendido (a_j: limite inferior de la ventana de tiempo del cliente j), el vehículo debe esperar hasta el tiempo a_j para poder iniciar el servicio a ese cliente.

   Reemplazando la ecuación **(4.16)** en la ecuación **(4.1)**, para este caso se obtiene que:

   $$te_k = a_j - ta_{jk} \quad [4.18]$$

   Reemplazando la ecuación **(4.18)** en la ecuación **(4.17)**, para est situación se obtiene que:

   $$w_{jk} = a_j \quad [4.19]$$

En este caso, coincide que, el tiempo de inicio de servicio del vehículo k en la locación del cliente j es igual al instante en el que dicho cliente acepta empezar a ser atendido (a_j: limite inferior de la ventana de tiempo del cliente j).

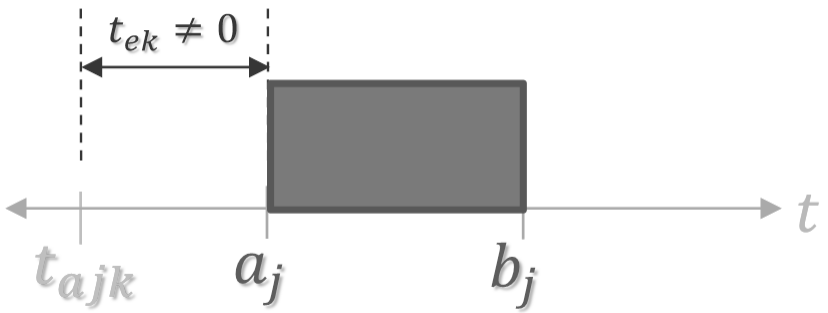

Figura 4.1: Caso 1

De acuerdo a la ecuación **(4.18)**, el tiempo que el vehículo k tiene que esperar en la locación del cliente j, antes de iniciar el servicio a dicho cliente, es igual al tiempo en el que ese cliente acepta empezar a ser atendido (a_j: Limite inferior de la ventana de tiempo del cliente j) menos el tiempo de llegada del vehículo k a la locación del cliente j, ta_{jk}.

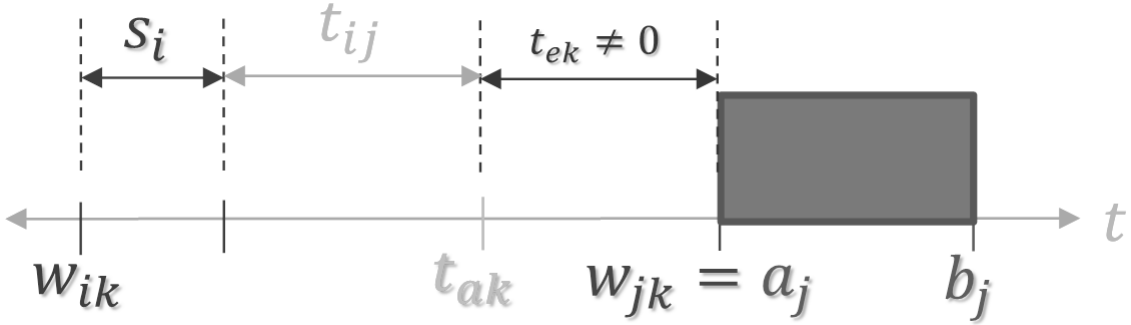

Figura 4.2: Tiempos para el Caso 1

Por lo tanto, el tiempo en el que el vehículo k inicia el servicio al cliente j, w_{jk}, es igual al tiempo de llegada del vehículo k a la locación del cliente j, ta_{jk}, más el tiempo de espera del vehículo k en la locación del cliente j, antes de que dicho cliente empiece a aceptar ser atendido.

2. Si $a_j \leq ta_{jk} \leq b_j$, entonces:

Si el tiempo de llegada del vehículo k a la locación del cliente j se encuentra entre los limites inferior y superior de la ventana de tiempo en la que de dicho cliente acepta ser atendido $[a_j, b_j]$, el vehículo debe iniciar inmediatamente el servicio a ese cliente. Por lo tanto, el tiempo de espera del vehículo k por llegadas dentro de la ventana de tiempo del cliente j es:

$$te_k = 0 \qquad [4.20]$$

Reemplazando la ecuación **(4.20)** en la ecuación **(4.17)**, para esta situación se obtiene que:

$$w_{jk} = ta_{jk} \qquad [4.21]$$

En este caso, el tiempo de llegada del vehículo k a la locación del cliente j es igual al tiempo de inicio de servicio del vehículo k al cliente j en su locación.

En las siguientes figuras se observan las tres posibles situaciones que se pueden presentar dentro del Caso 2.

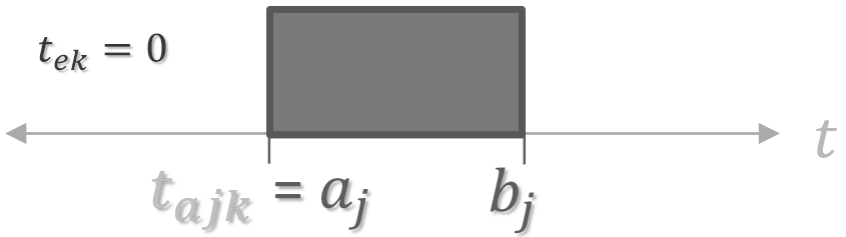

Figura 4.3: Caso 2a

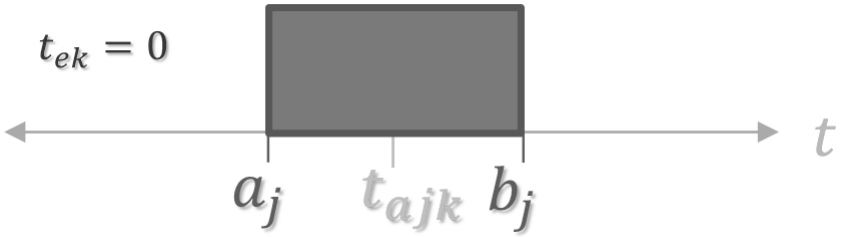

Figura 4.4: Caso 2b

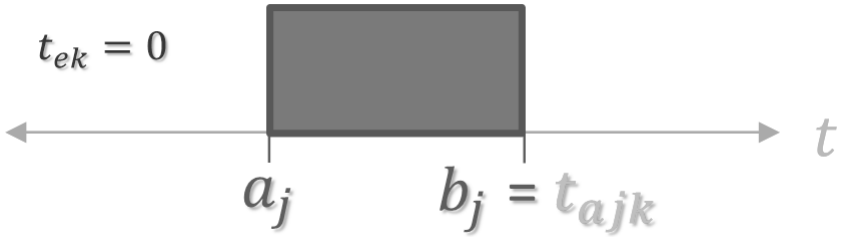

Figura 4.5: Caso 2c

Para esta situación, el tiempo de espera del vehículo k al llegar a la locación del cliente j es igual a cero, $te_k = 0$, debido a que este vehículo llegó justo en el momento en el que se abre la ventana de tiempo para ese cliente, es decir, en el instante a_j, (Ver **Figura** C.**11**), o cuando esta ventana ya se encontraba abierta, es decir, entre $[a_j, b_j]$, (Ver **Figura 4.6**), o justo antes de que se cerrara la ventana de tiempo de ese cliente,

en el instante b_j, (Ver **Figura 4.7**), por lo tanto, el vehículo debe iniciar su servicio de inmediato.

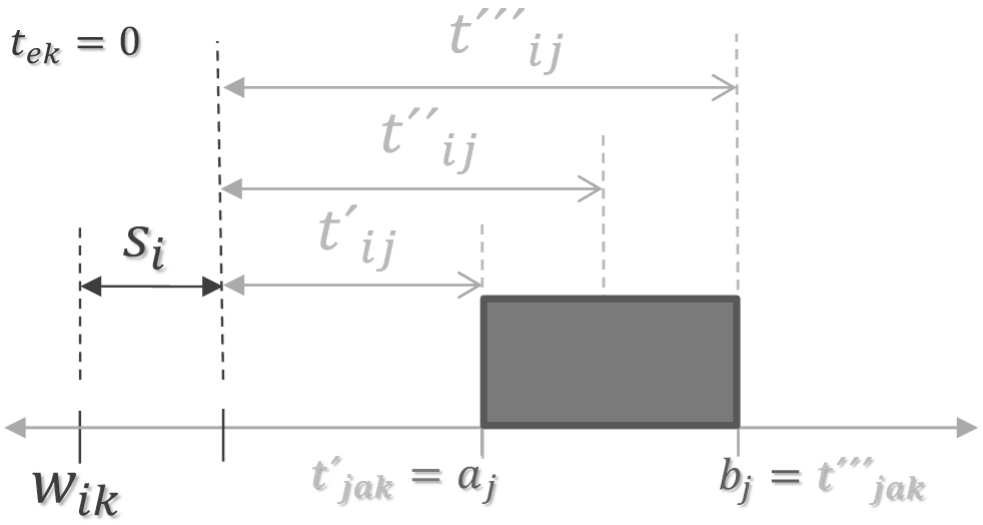

Figura 4.6: Tiempos para el Caso 2

3. Si $ta_{jk} > b_j$, entonces:

   La solución es infactible debido a que el vehículo k viola la restricción de la ventana de tiempo del cliente j llegando después del tiempo más tarde en el que dicho cliente acepta ser atendido. Es decir, después del instante b_j.

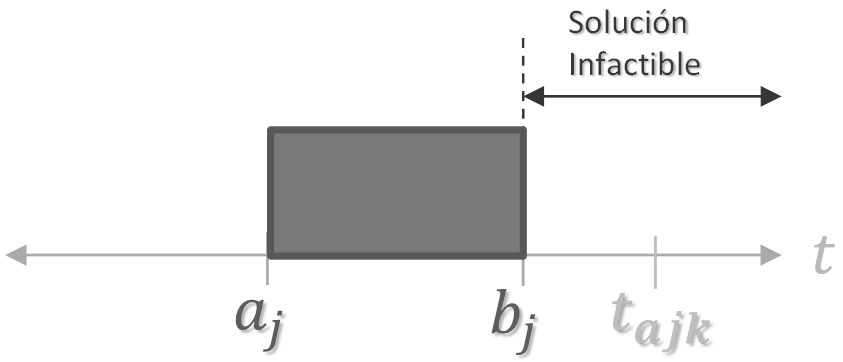

Figura 4.7: Caso 3

4.3.3. Construcción de Soluciones Iniciales

Como se mencionó anteriormente, esta investigación modifica la Heurística de Inserción I1, adaptándola a instancias con ubicaciones reales y a rutas sometidas a congestión vehicular, con el fin de generar soluciones de buena calidad. Teniendo en cuenta el número limitado de peticiones que se pueden realizar al servidor de Google por día en la versión no paga, máximo 100.000, las instancias de prueba deben ser pequeñas, y no se utilizarán las tres posibilidades diferentes de escoger al cliente semilla y las cuatro configuraciones diferentes de parámetros para el procedimiento clásico de inserción I1 que aparecen en (**?**), que resultan en 12 soluciones iniciales diferentes. Sino que se utilizará solamente una de las posibilidades de escoger al cliente semilla y una de las configuraciones de parámetros para la construcción de la solución, ver: ecuación **(4.31)**.

Para iniciar una ruta, se selecciona al cliente semilla, que será el cliente sin enrutar más alejado del depósito central. En un principio, cada ruta inicial contendrá sólo al depósito central como origen de la ruta, al cliente semilla y al depósito central como destino final de la ruta, tal como se observa a continuación: [0 $Semilla$ 0]. Esta ruta parcialmente construida es luego rellenada con clientes sin enrutar.

La modificación que se realizó al algoritmo original fue que, de ser factible la inserción del cliente, en cuanto al tiempo se realiza lo siguiente:

Para entender como funciona RuGle, en toda la línea de tiempo que hay entre un nodo i y un nodo j, se hacen varios cortes que se denominarán ranuras de tiempo (RT). Debido a que Google Maps hace los cálculos en intervalos de 1 hora, cada RT tendrá un intervalo igual. Adicionalmente, cada RT tendrá un valor t_{ij}, del nodo i al nodo j, dado que para cada una de ellas el tiempo de viaje será diferente, se toma la ranura con el menor tiempo, de esta manera se puede obtener un menor tiempo de viaje total. Por ejemplo:

El cliente j a visitar tiene una ventana de tiempo de [12pm-4pm], las ranuras de tiempo para ese caso serían:

Tabla 4.2: Ranuras de Tiempo de 1 Hora para una Ventana de Tiempo de 12pm-4pm

12PM-1PM	1PM-2PM	2PM-3PM	3PM-4PM

De **12pm-1pm**, el recorrido tarda unos 40 minutos aproximadamente.
De **1pm-2pm**, el recorrido tarda unos 35 minutos.
De **2pm-3pm**, el recorrido tarda unos 30 minutos.
De **3pm-4pm**, el recorrido tarda unos 20 minutos.

De estas 4 RT, RuGle busca el menor tiempo de viaje posible, y ese es el que finalmente toma para calcular la mejor ubicación de un nodo en la ruta que se está construyendo. La heurística original de Inserción I1, no tiene en cuenta esto, debido a que trata la distancia y el tiempo como si fuesen lo mismo, además, no contempla la congestión vehicular al calcular el tiempo de viaje entre los nodos i y j, la cual varía dependiendo del horario. RuGle hará la petición al servidor de Google para conocer el tiempo de viaje entre dos nodos en busca del

mejor instante en el que el tiempo entre ellos sea menor, mientras que la heurística original sacará estos datos de una matriz estática de distancia/tiempo. Del ejemplo, queda claro que para cada instante se tiene un tiempo de viaje distinto entre dos nodos y no un tiempo de viaje constante entre un nodo i y un nodo j, y que estos tiempos variarán dependiendo de cuando se haga la consulta, ya que Google maneja estimados, lo que hace un poco más cercanos a la realidad los resultados que RuGle ofrece.

Dos medidas de costo $c_1(i,u,j)$ y $c_2(i,u,j)$ son empleadas para identificar al cliente a ser insertado dentro de la ruta parcialmente construida y para encontrar la mejor ubicación factible entre nodos consecutivos i y j. El proceso de inserción consiste de tres pasos. Al principio, el mínimo costo de inserción factible $c_1^*(i,u,j)$ para cada cliente sin enrutar u es computado. El costo c_1 por insertar al cliente u entre dos nodos adyacentes i y j es calculado como:

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) \qquad [4.22]$$

Con:

$$\alpha_1 + \alpha_2 = 1 \qquad [4.23]$$

$$\alpha_1 \geq 0 \qquad [4.24]$$

$$\alpha_2 \geq 0 \qquad [4.25]$$

La distancia adicional que un vehículo tiene que aceptar cuando sirve al cliente u en la misma ruta que a j es representado por $c_{11}(i,u,j)$. La componente $c_{12}(i,u,j)$ es la moodifi-

cación del inicio del tiempo de servicio en el cliente j cuando u es insertada antes que j en la ruta. Estas dos componentes son calculadas como:

$$c_{11}(i,u,j) = t_{iu} + t_{uj} - \mu t_{ij}, \qquad [4.26]$$

$$\mu \geq 0 \qquad [4.27]$$

y

$$c_{12}(i,u,j) = w_{ujk} - w_{jk} \qquad [4.28]$$

Donde : t_{iu}, t_{uj} y t_{ij} representa el tiempo de viaje entre los clientes i y u, u y j, e i y j, respectivamente.

La mayoría de autores, para éste tipo de problemas, emplean la distancia y tiempo indistintamente. Pero para efectos de esta investigación, consideraremos la distancia y el tiempo de viaje cómo diferentes. Además, en cuanto al tiempo de viaje, este se considerará con congestión y sin congestión vehicular.

El tiempo de inicio de servicio en el cliente j es denotado por w_{jk}, y w_{ujk} representa el nuevo tiempo de inicio de servicio en el cliente j con u insertado después de j.

Segundo, el cliente sin enrutar u^* es seleccionado entre los clientes que pueden ser factiblemente insertados dentro de la ruta.

Este cliente maximiza el criterio $c_2(i,u,j)$, la medida generalizada de ahorro (generalized saving measure), que es obtenida de:

$$c_2(i,u,j) = \lambda t_{0u} - c_1^*(i,u,j), \qquad [4.29]$$

$$\lambda \geq 0 \qquad [4.30]$$

En el último paso, el cliente u^* es insertado en su mejor posición. La inserción de clientes es ejecutada mientras que la capacidad y las restricciones de la ventana de tiempo no sean violadas.

Si quedan clientes sin enrutar que no pudieron ser factiblemente enrutados, una nueva ruta es inicializada y el proceso de inserción es repetido. El procedimiento finaliza cuando cada cliente es asignado a un vehículo

Se utilizó la siguiente configuración de parámetros para el procedimiento de inserción:

$$(\alpha_1, \alpha_2, \mu, \lambda) = (1, 0, 1, 1) \qquad [4.31]$$

Se denota al número de vehículos en la solución como K, y la distancia total recorrida como d. La solución es denotada por:

$$S = R_1, ..., R_p, \qquad [4.32]$$

Donde:

$$R_p = (V_{R_p,1}, V_{R_p,2}, ..., V_{R_p,n_p}) \qquad [4.33]$$

Es el conjunto de clientes atendidos por el vehículo p. La secuencia de los clientes en la ruta R_p es equivalente al orden en el que serán atendidos por el vehículo. En el Apéndice A, se muestra el pseudocódigo de RuGle.

4.4. COMPLEJIDAD COMPUTACIONAL DEL CÓDIGO DE RuGle

De lo encontrado en las subrutinas y basado en el algoritmo del **Apendice A**, la ecuación de su complejidad se encuentra dada por $SSS(n) = \Theta(n^4)$

Notas:

inicializar_ ruta: es una función que recibe como parámetro el criterio que se va a utilizar para seleccionar al que será el cliente semilla, en este caso fue el del cliente sin enrutar más alejado del depósito central. El segundo parámetro que recibe es la capacidad que tiene el vehículo. La complejidad de esta función es n.

insertar_ nodo: es una función que recibe como parámetro una ruta, un nodo y la posición, la función inserta el nodo en la posición indicada a la ruta, se calculan los nuevos valores de la ruta si es que es una solución factible, de otro modo la función retorna falso. La complejidad de esta función es n.

calcular_ $c2$: es una función que calcula el valor de $c_2(i, u, j) = \lambda t_{0u} - c_1^*(i, u, j), \lambda \geq 0$, su complejidad es de 1.

Restricciones de las peticiones no pagas de Google Maps: La llave gratis que Google proporciona solo permiten un número máximo de 100,000 peticiones a su servidor cada 24 horas.

CAPÍTULO 5

VERIFICACIÓN DE LA HERRAMIENTA

En este capítulo se presentan los resultados obtenidos luego de probar RuGle en dos instancias pequeñas (construidas de acuerdo a lo descrito en el **Apéndice B: "Manual de Usuario de RuGle"**, **Sección B.1)**, teniendo en cuenta y no el tráfico.

5.1. RESULTADOS OBTENIDOS CON LA INSTANCIA DE PRUEBA n4

Los nodos que conforman la instancia de prueba son ubicaciones reales en la ciudad de Barranquilla, que se encuentran geográficamente dispersas unas de otras. Cada nodo cuenta con la dirección, que permite ubicarlo en el mapa digital, demanda, ventana de tiempo y tiempo de servicio. La siguiente imagen muestra la instancia de prueba construida:

Tabla 5.1: Instancia de Prueba con 4 Nodos

ID	ADDRESSS	DEMAND	READY_TIME	DUE_DATE	SERVICE_TIME
0	Calle 70 #43-130, Barranquilla, Atlantico, Colombia	0	6:00	18:00	0
1	Calle 90 #56-56, Barranquilla, Atlantico, Colombia	200	6:00	7:00	900
2	Carrera 70 # 82-82, Barranquilla, Atlantico, Colombia	20	8:00	10:00	900
3	Calle 79 # 60-60, Barranquilla, Atlantico, Colombia	150	6:00	10:00	900

Luego de construida la instancia, se prueba el VRPHTW con ella. Empleando la herramienta desarrollada (RuGle), se obtuvieron los siguientes resultados:

```
1.      [  0  3  2  0  ]
        Time: 8229.0 s
        Time with traffic: 8211.0 s
        Distance: 8317.0 m
        Waiting time: 4576.0 s
        Waiting time with traffic: 4955.0 s
        Services time: 1800 s
        Arrival time: 2016-10-27 08:17:09 UTC
        Arrival time with traffic: 2016-10-27 08:16:51 UTC
        Demand: 170
2.      [  0  1  0  ]
        Time: 2717.0 s
        Time with traffic: 2373.0 s
        Distance: 8128.0 m
        Waiting time: 0 s
        Waiting time with traffic: 0 s
        Services time: 900 s
        Arrival time: 2016-10-27 06:45:17 UTC
        Arrival time with traffic: 2016-10-27 06:39:33 UTC
        Demand: 200
```

Figura 5.1: Resultados Obtenidos con RuGle para una Instancia de 4 Nodos

De los resultados se observa que, para poder atender a los 3 clientes que conforman la instancia objeto de estudio, la solución propuesta por RuGle emplea 2 vehículos, la distancia total recorrida por la flota vehicular es de 16,4 Km, el tiempo total de viaje de la flota sin tener en cuenta el tráfico es de 227,7 minutos y teniendo en cuenta el tráfico es de 176,4 min, el tiempo total de espera sin tráfico es de 76,2 min y el tiempo total de espera con tráfico es de 82,6 min. Para el tiempo de llegada de toda la flota al depósito central, sin tener en cuenta el tráfico, tomamos el tiempo del último vehículo en llegar al depósito al fnalizar su recorrido, en este caso fue: 2016-10-27 08:17:09 UTC. Para el tiempo de llegada de toda la flota al depósito central, teniendo en cuenta el tráfico, tomamos el tiempo del último vehículo

en llegar al al depósito al fnalizar su recorrido, en este caso fue a las 2016-10-27 08:16:51 UTC.

Número Total de Vehículos: 2

Distancia Total Recorrida: $16445m\ \ (16,4Km)$

Tiempo Total de Viaje, sin Tráfico: $13663s\ \ (227,7min)$

Tiempo Total de Viaje, con Tráfico: $10584s\ \ (176,4min)$

Tiempo Total de Espera, sin Tráfico: $4576s\ \ (76,2min)$

Tiempo Total de Espera, con Tráfico: $4955s\ \ (82,6min)$

Tiempo Total de Servicio: $2700s\ \ (45min)$

Tiempo de Llegada sin Tráfico de Toda la Flota: 2016-10-27 08:17:09 UTC

Tiempo de Llegada con Tráfico de Toda la Flota: 2016-10-27 08:16:51 UTC

Demanda Total: 370

Se observa que, la instancia de prueba cuenta con ventanas de tiempo mixtas, compuesta por ventanas estrechas (de apenas 1 y 2 horas) y amplias (de 4 horas) para los clientes que la conforman. Cómo las ventanas de tiempo de los clientes tienen poca flexibilidad de scheduling (horizonte de planificación corto o ventanas de tiempo muy estrechas), sólo es posible atender a pocos clientes en cada ruta. Por otro lado, las instancias de prueba con mayor flexibilidad de scheduling permitirián atender a más clientes por vehículo. Por lo tanto, aunque la instancia conste de sólo 3 clientes y la capacidad total de cada vehículo aún no haya sido ocupada por completo, debido a que la mayoría de las ventanas de tiempo de estos presenta un horizonte de planificación corto y la suma de la demanda total de todos los clientes excede la capacidad de un sólo vehículo, es necesario un vehículo adicional para poder atenderlos a todos.

$k_{min} = [(\sum_{i \in N} d_i)/C]$

$k_{min} = [370/200]$

$k_{min} = 1{,}85 \approx 2$

Aplicando la ecuación **(4.15)**, confirmamos que, para este caso ya no es posible seguir minimizando el número total de vehículos por debajo de 2.

La solución hallada, gráficamente luce de la siguiente forma:

Figura 5.2: Solución Gráfica Obtenida con RuGle para una Instancia de 4 Nodos

En esta instancia, la solución hallada presenta tiempos de espera para una de las dos rutas que la conforman. Esto se debe a que los vehículos llegaron antes de que la ventana de tiempo, de uno o más clientes de dicha ruta, se abriera.,En el caso del *cliente*1 y el *cliente*3, ellos empiezan a aceptar ser atendidos a partir de las 6 : 00, la misma hora a la que parten los vehículos del depósito central, mientras que el *cliente*2 sólo acepta ser atendido a partir de las 8 : 00, si un vehículo llega antes de esa hora a la ubicación del *cliente*2, debe esperar. Por lo tanto, el tiempo de inicio de servicio del *vehículo*1 en el *cliente*2 es igual al límite inferior de su ventana de tiempo, es decir, 8 : 00, tal cómo se describe en el **Caso 1**, **Sección 4.3.2.**. En cuanto al *cliente*3, este es el primer cliente visitado por el *vehículo*1. Este vehículo llega

a la ubicación del cliente cuando la ventana de tiempo de este ya se encuentra abierta, para esta situación, el tiempo de espera es 0 y el tiempo de inicio de servicio y de llegada del *vehículo*1 al *cliente*3 coinciden, tal cómo se muestra en el **Caso 2**, **Sección 4.3.2.**. Como el *cliente*1, tiene una ventana de tiempo estrecha, de tan sólo una hora $[6:00-7:00]$ y una demanda de 200 unidades, el equivalente a la capacidad de carga total de un vehículo de la flota, el vehículo que lo atiende no puede visitar a otros clientes en su ruta, ya que la demanda de este es igual a la capacidad máxima del vehículo. Todos los vehículos parten del depósito central a las 6 : 00, por lo tanto, el vehículo asignado a esa ruta, si tiene la posibilidad de llegar en la siguiente hora al *cliente*1, teniendo en cuenta y no la congestión, no tendrá que esperar tiempo alguno debido a que la ventana de este ya se encuentra abierta, por lo tanto, el tiempo de espera para esta ruta es 0. Para esta situación, el tiempo de inicio de servicio y de llegada del *vehículo*2 al *cliente*1 también coinciden, tal cómo se describe en el **Caso 2**, **Sección 4.3.2.**

5.2. RESULTADOS OBTENIDOS CON LA INSTANCIA DE PRUEBA n10

Los nodos que conforman la instancia de prueba son ubicaciones reales en la ciudad de Barranquilla, que se encuentran geográficamente dispersas unas de otras. Cada nodo cuenta con la dirección que permite ubicarlo en el mapa digital, demanda, ventana de tiempo y tiempo de servicio. La siguiente imagen muestra la instancia de prueba construida:

Luego de construida la instancia, se prueba el VRPHTW con ella. Empleando la herramienta desarrollada (RuGle), se obtuvieron los siguientes resultados:

Tabla 5.2: Instancia de Prueba con 10 Nodos

ID	ADDRESSS	DEMAND	READY_TIME	DUE_DATE	SERVICE_TIME
0	Calle 70 #43-130, Barranquilla, Atlantico, Colombia	0	6:00	18:00	0
1	Calle 90 #56-56, Barranquilla, Atlantico, Colombia	45	6:00	7:00	900
2	Carrera 70 # 82-82, Barranquilla, Atlantico, Colombia	20	8:00	10:00	900
3	Calle 79 # 60-60, Barranquilla, Atlantico, Colombia	50	6:00	10:00	900
4	Cra 42 # 56-11, Barranquilla, Atlantico, Colombia	20	6:00	7:00	900
5	Calle 53d # 31-31, Barranquilla, Atlantico, Colombia	50	8:00	10:00	900
6	Calle 45 # 35-35, Barranquilla, Atlantico, Colombia	45	6:00	10:00	900
7	Calle 72 # 65-65, Barranquilla, Atlantico, Colombia	20	6:00	7:00	900
8	Carrera 71 # 76-76, Barranquilla, Atlantico, Colombia	45	8:00	10:00	900
9	Carrera 82 # 58-157, Barranquilla, Atlantico, Colombia	50	6:00	10:00	900

```
1.      [  0  9  6  4  5  2  0  ]
        Time: 8568.0 s
        Time with traffic: 7751.0 s
        Distance: 22174.0 m
        Waiting time: 0 s
        Waiting time with traffic: 0 s
        Services time: 4500 s
        Arrival time: 2016-10-27 08:22:48 UTC
        Arrival time with traffic: 2016-10-27 08:09:11 UTC
        Demand: 185
2.      [  0  1  3  7  8  0  ]
        Time: 6713.0 s
        Time with traffic: 6130.0 s
        Distance: 16480.0 m
        Waiting time: 0 s
        Waiting time with traffic: 0 s
        Services time: 3600 s
        Arrival time: 2016-10-27 07:51:53 UTC
        Arrival time with traffic: 2016-10-27 07:42:10 UTC
        Demand: 160
```

Figura 5.3: Resultados Obtenidos con RuGle para una Instancia de 10 Nodos

De los resultados se observa que, para poder atender a los 9 clientes que conforman la instancia objeto de estudio, la solución popuesta por RuGle emplea 2 vehículos, la distancia total recorrida por la flota vehicular es de 38,7 Km, el tiempo total de viaje de la flota sin tener en cuenta el tráfico es de 254,7 minutos y teniendo en cuenta el tráfico es de 231,4min, el tiempo total de espera con y sin tráfico es de 0s. Para el tiempo de llegada de toda la flota al depósito central, sin tener en cuenta el tráfico, tomamos el tiempo del último vehículo en llegar al al depósito al fnalizar su recorrido, en este caso fue a las 2016-10-27 08:22:48 UTC.

Para el tiempo de llegada de toda la flota al depósito central, teniendo en cuenta el tráfico, tomamos el tiempo del último vehículo en llegar al al depósito al fnalizar su recorrido, en este caso fue a las 2016-10-27 08:09:11 UTC.

Número Total de Vehículos: 2

Distancia Total Recorrida: $38654m\ \ (38,7Km)$

Tiempo Total de Viaje, sin Tráfico: $15281s\ \ (254,7min)$

Tiempo Total de Viaje, con Tráfico: $13881s\ \ (231,4min)$

Tiempo Total de Espera, sin Tráfico: $0s$

Tiempo Total de Espera, con Tráfico: $0s$

Tiempo Total de Servicio: $8100s\ \ (135min)$

Tiempo de Llegada sin Tráfico de Toda la Flota: 2016-10-27 08:22:48 UTC

Tiempo de Llegada con Tráfico de Toda la Flota: 2016-10-27 08:09:11 UTC

Demanda Total: 345

Se observa que la instancia de prueba cuenta con ventanas de tiempo mixtas, compuesta por ventanas estrechas (de apenas 1 y 2 horas) y amplias (de 4 horas), para los clientes que la conforman. Cómo las ventanas de tiempo de los clientes son más variadas en esta instancia, presentando más flexibilidad de scheduling (ventanas de tiempo más amplias), por lo cual, es posible atender a más clientes por ruta. Por otro lado, aunque la instancia conste de sólo 9 clientes, se necesitarán de mínimo 2 vehículos para poder cumplir con la demanda de todos ellos debido a la capacidad limitada de los vehículos.

$k_{min} = [(\sum_{i \in N} d_i)/C]$

$k_{min} = [345/200]$

$k_{min} = 1{,}7 \approx 2$

Aplicando la ecuación **(4.15)**, confirmamos que, para este caso ya no es posible seguir minimizando el número total de vehículos por debajo de 2.

La solución hallada, gráficamente luce de la siguiente forma:

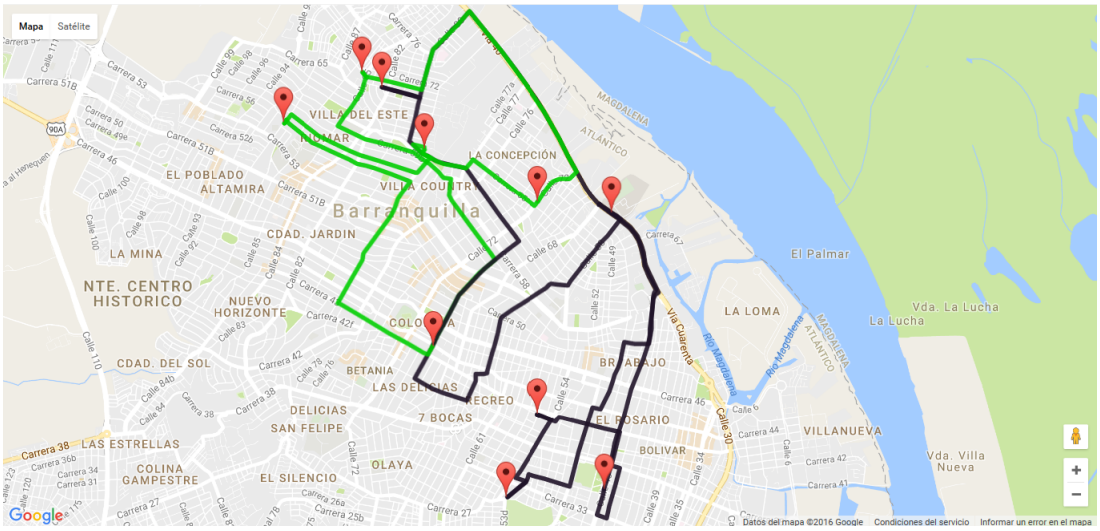

Figura 5.4: Solución Gráfica Obtenida con RuGle para una Instancia de 10 Nodos

En esta instancia, se observa que, para la solución hallada, las dos rutas que la conforman no presentan tiempo de espera, debido a que todos los vehículos de la flota llegaron a la ubicación de los clientes cuando las ventanas de tiempo de estos ya se encontraban abiertas, por lo que, el tiempo de llegada de los vehículos a los clientes, coincidió con el tiempo de inicio de servicio a esos clientes, tal cómo se describe en el **Caso 2**, **Sección 4.3.2.**.

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se presentan las conclusiones de la presente investigación y algunos de los posibles trabajo futuros que puedan continuar desarrollándose como resultado de este proyecto de grado.

6.1. CONCLUSIONES

Durante la presente investigación, se logró desarrollar una alternativa para resolver un problema combinatorio multiobjetivo, el VRPHTW sometido a rutas con congestión, a través de una herramienta que incorpora elementos de solución clásicos, tales como: la heurística de inserción I1 **?** y plataformas de ultima tecnología como Google Maps. Uno de los principales desafíos que se tuvieron que enfrentar, fueron las limitaciones encontradas al emplear la llave gratuita que Google proporciona, la cual, Ãžnicamente permite un máximo de $100,000$ peticiones a su servidor cada 24 horas, por lo que las instancias de prueba a construir no podían ser demasiado grandes ni se podía incrementar el nivel de complejidad del modelo matemático a resolver. Por lo tanto, se buscó una forma de tener en cuenta la congestión sin tener que incorporar ecuaciones para calcular el flujo vehicular al modelo, fue así como se decidió utilizar la API de Google, ya que RuGle simplemente realiza una petición a esta

interfaz para conocer en cada instante el estado de las vías que comunican a uno nodo con otro, mostrando así, el gran potencial de la plataforma online de Google en el estudio y planteamiento de problemas de enrutamiento de vehículos cada vez más realistas, y a un menor costo computacional, debido a que se reduce el número de cálculos matemáticos a realizar y sin incorporar ecuaciones para estimar el flujo vehicular, reemplazándolos por peticiones a este servidor .

RuGle permite construir soluciones, junto con sus respectivos grafos, para un VRPHTW multiobjetivo sometido a rutas con congestión, empleando ubicaciones reales en cualquier zona del planeta, sin limitarse únicamente a la ciudad de Barranquilla, gracias a que está construida sobre la plataforma de Google Maps, la cual abarca todo el globo térraqueo con mapas digitales actualizados de cualquier zona del mundo.

Durante el análisis de los resultados obtenidos con RuGle, se encontró que, en algunas instancias puede ocurrir que los tiempos de viaje con congestión de un nodo i a un nodo j, sean menores que los tiempos de viaje sin congestión entre ellos, pero hay que tener en cuenta que Google Maps maneja estimaciones, por lo tanto, los resultados a veces pueden quedar por encima o por debajo del promedio al utilizar la herramienta propuesta.

En los resultados arrojados por RuGle, también se encontró que ventanas de tiempo estrechas, ocasionan que un vehículo sea capaz de visitar menos clientes en su ruta. Mientras que, ventanas de tiempo amplias le permitirán al vehículo visitar más clientes en su ruta, siempre y cuando su capacidad se lo permita. Porque también se presentaron casos, como en la instancia de prueba $n4$, en las que la demanda de un sólo cliente igualaba la capacidad total de los vehículos de la flota homogénea, por lo tanto, el vehículo encargado de atender a ese cliente no podría visitar a otros en su ruta, ya que terminaría violando la restricción de capacidad.

Por otro lado, RuGle siempre busca el mejor instante en una linea de tiempo para que el vehículo k parta del nodo i con destino al nodo j en el menor tiempo de viaje posible, teniendo en cuenta y no teniendo en cuenta la congestión vehícular. Pero el tiempo de espera puede verse incrementado, por ejemplo: al mejorar el tiempo de viaje o la distancia recorrida, o ambos, se pueden presentar llegadas adelantadas, lo cual se traduce en varios minutos de espera para el vehículo k en la locación del cliente. Aquí queda claro que, al contar con varios objetivos, estos en muchas ocasiones entrarán en conflicto.

Al analizar los resultados obtenidos con ambas instancias, encontramos que la $n4$ es la única, de las dos instancias construidas, cuya solución genera tiempos de espera. Esto se debe a que, uno de los vehículos de la flota llegó antes de que la ventana de tiempo de uno de los clientes en su ruta se abriera. Mientras que en la instancia $n9$, los vehículos no presentaron tiempo de espera alguno, debido a que llegaron a la ubicación de los clientes justo cuando las ventanas de tiempo de estos ya se encontraban abiertas.

Finalmente, gracias en parte a que RuGle fue desarrollado en un lenguaje de programación de código abierto, cualquiera que desee iniciarse en el campo del problema de enrutamiento de vehículos con ventanas de tiempo, encontrará muy útil esta herramienta autodidáctica como punto de partida para estudiar y desarrollar algoritmos de optimización combinatoria cada vez más complejos y eficientes.

6.2. TRABAJOS FUTUROS

A continuación se presentan algunos trabajos futuros que pueden desarrollarse como resultado de esta investigación o que, por exceder el alcance de esta tesis, no han podido ser tratados con la suficiente profundidad. Además, se sugieren algunos desarrollos específicos para apoyar y mejorar el modelo y metodología propuestos.

Luego de estudiar el tema de enrutamiento de vehículos con ventanas de tiempo duras, es recomendable investigar las aplicaciones que este tiene en la realidad. A lo largo de esta tesis se ha explicado la aplicabilidad y los beneficios que otorgaría implementar sistemas inteligentes que hallen rutas óptimas considerando las restricciones del modelo. Sería recomendable tomar este sistema de prueba como núcleo para sistemas más complejos (a nivel funcional).

Entre los posibles trabajos futuros se destacan:

Continuar resaltando las amplias posibilidades que puede llegar a tener el uso de los mapas digitales en problemas de enrutamiento de vehículos, y que RuGle sirva como inspiración para un próximo proyecto mucho más ambicioso, que aborde los VRPTW de manera más real. Para lo cual se sugiere como futura investigación, emplear el modelo matemático propuesto pero adicionándole una tercera función objetivo: la función de penalizaciones, e imponerle el límite máximo de llegadas adelantadas y llegadas tardías en cada cliente, con el fin de prevenir tiempos de espera y retraso muy largos.

Al permitir las violaciones de las ventanas de tiempo, pero asociadas a una penalización, se cuenta con más posibilidades para elegir el próximo cliente a visitar. Especialmente en los casos en el que las violaciones causan el uso de menos vehículos o disminuciones sustanciales en la distancia total recorrida y menor tiempo de espera, incurrir en penalizaciones puede ser aceptables.

Es importante que, para prevenir largos tiempos de espera y retraso, se impongan limites máximos de llegadas adelantadas y llegadas tardías en cada cliente. Debido a que resulta poco práctico permitir tiempos de espera sin límite en la locación del cliente, aunque se asigne una penalización, ya que ésta probablemente podría llegar a ser infinita después de un cierto tiempo.

En el VRPHTW, un vehículo puede llegar a la locación del cliente antes del límite inferior de la ventana de tiempo a_i, pero debe esperar, sin recibir penalización alguna, hasta que se abra la ventana de tiempo inferior para poder iniciar el servicio a ese cliente. Cualquier visita o servicio después del límite superior de la ventana de tiempo b_i produce soluciones infactibles. Esto evidencia las debilidades del VRPHTW al aplicarse a situaciones de la vida real, donde gestionar los vehículos para atender a todos los clientes dentro de las ventanas de tiempo es muy difícil de lograr en la práctica, y es muy probable que se incrementen los costos debido al uso de muchos vehículos con el fin de cumplir con todas estas. Para evitar estos inconvenientes, **?** relajaron parcialmente las ventanas de tiempo introduciendo penalizaciones dependientes del tiempo, por llegadas tardías al problema. El problema es descrito como el VRPSSTW (The VRP With Semi-Soft Time Windows: Problema de Enrutamiento de Vehículos con Ventanas de Tiempo Semi Suaves). En el VRPSSTW, se le permite a los vehículos visitar y servir a los clientes después del límite superior de la ventana de tiempo establecido b_i, sin embargo, una penalización dependiente del tiempo, por llegadas tardías, debe aplicarse por presentarse demoras en el servicio. Si llega temprano a la locación del cliente, el vehículo aún debe esperar para iniciar el servicio hasta a_i, sin recibir por ello penalización alguna. **?** y **?** también introdujeron el máximo tiempo permitido de llegadas tardías en cada cliente a ser visitado (representado por $b_i{}'$) considerando concesiones entre costos fijos y penalizaciones por llegadas tardías.

? muestran que el VRPSSTW, comparado con el VRPHTW, puede minimizar el número de vehículos en la solución. Por lo tanto, el Costo Total puede ser reducido consecuentemente. Adicionalmente, el tiempo total de espera de los vehículos, aunque no es considerado como un objetivo principal, también es minimizado. Sin embargo, se encontró, que tanto el VRPHTW como el VRPSSTW, generan tiempos de espera significativos en la práctica. Para tratar de resolver este inconveniente, el VRPSTW incluye, a parte de las penalizaciones por llegadas tardías, las penalizaciones por llegadas adelantadas:

- Penalizaciones para el VRPSTW:

    - Penalizaciones por Llegadas Tardías: se incurre en estas penalizaciones cuando el vehículo llega después de la ventana superior b_i establecida por el cliente e inicia su servicio de inmediato, sin ningún tiempo de espera.
    - Penalizaciones por Llegadas Adelantadas: se incurre en estas penalizaciones cuando el vehículo llega antes de la ventana inferior a_i establecida por el cliente e inicia su servicio de inmediato, sin ningún tiempo de espera.

Con estas penalizaciones por llegadas adelantadas, se le da la oportunidad al vehículo de minimizar el tiempo de espera en la locación del cliente, iniciando el servicio antes del tiempo pactado con este, pero asociada a una penalización. En este problema no sólo se debe apuntar a minimizar el costo total y las penalizaciones por llegadas adelantadas o tardías, sino también, de manera simultanea, el tiempo total de espera.

Sin embargo, permitir tiempos de espera sin límite en la locación del cliente, aunque se asigne una penalización, resulta siempre poco práctico y la penalización correspondiente probablemente podría llegar a ser infinita después de un cierto tiempo. Por lo tanto, es importante que, para prevenir largos tiempos de espera y retraso, se impongan limites máximos de llegadas adelantadas y llegadas tardías en cada cliente. Durante la presente investigación se encontró que **?** son los primeros en definir claramente estos limites, ellos proponen la formulación para determinar el limite máximo de llegadas adelantadas (espera) y llegadas tardías de un vehículo para llegar a cada cliente.

Cabe resaltar que, para el VRPTW y sus variantes en general (VRPHTW, VRPSSTW y VRPSTW), si un vehículo llega dentro de la ventana de tiempo establecida por el cliente $[a_i, b_i]$, al vehículo no le es permitido esperar, debe iniciar el servicio de inmediato, sin recibir penalización alguna, porqué no esta violando las restricciones temporales de servicio. Y para el VRPSSTW y el VRPSTW, si el vehículo llega después de la ventana superior b_i establecida

por el cliente, tampoco le es permitido esperar bajo ninguna circunstancia, debe iniciar su servicio de inmediato, pero a cambio recibirá una penalización por violar la ventana de tiempo.

Desde la perspectiva de los problemas de optimización combinatoria, el VRPSTW es mucho más difícil de resolver hasta la optimalidad que el VRPHTW y el VRPSSTW. El principal inconveniente al resolver el VRPSTW es que las penalizaciones dependientes del tiempo, tanto por llegadas adelantadas como por llegadas tardías, son consideradas.

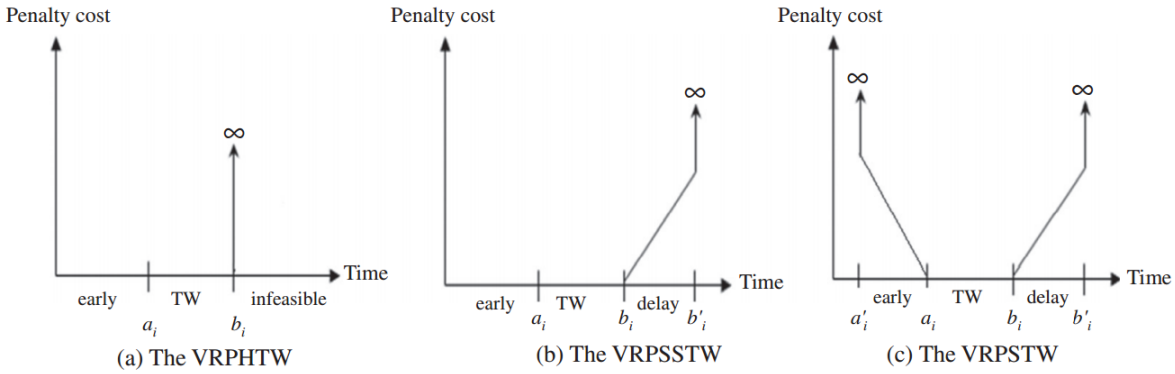

Figura 6.1: Función de Penalización de Llegadas de las Tres Variantes del VRP con Ventanas de Tiempo
Fuente: Narath Bhusiri, Ali Gul Qureshi and Eiichi Taniguchi, 2014

En general, los tipos de penalizaciones pueden variar de un cliente a otro, dependiendo de la importancia, prioridad o criticalidad.

Por lo tanto se propone resolver tanto el VRPSTW como el VRPSSTW, partiendo del modelo propuesto, como parte de futuras investigaciones, para lo cual muy probablemente se tendrá que evaluar la posibilidad de adquirir la versión paga de Google Maps debido a las limitaciones en cuanto al número de peticiones que se le pueden realizar por día al servidor con la versión gratuita.

APÉNDICES

APÉNDICE A

PESUDOCÓDIGO DE RuGle

Cargar nodos // Complejidad n

rutas = []

$nodos\_no\_visitados\_copia \leftarrow nodos\_no\_visitados\_original$

$ruta\_original \leftarrow inicializar\_ruta(semilla, capacidad)$ // Complejidad n

hacer

$no\_visitados \leftarrow nodos\_no\_visitados\_original$

$nueva\_ruta \leftarrow nulo$

$nodo\_agregado \leftarrow nulo$

$máximo\_c2 \leftarrow -infinito$

Para cada nodo **en** $no\_visitados$

para $posición \leftarrow 1$ **hasta** $ruta\_original.tamaño$

$ruta\_alternativa \leftarrow ruta\_original$

Si $insertar\_nodo(ruta\_alternativa, nodo, posición)$ // Complejidad n

$c2 \leftarrow calcular\_c2(ruta\_alternativa, ruta\_original, nodo, posición, criterio)$ // Complejidad 1

Si $c2 \geq máximo\_c2$

$máximo\_c2 = c2$

$nueva\_ruta = ruta\_alternativa$

$nodo\_agregado = nodo$

Fin si

Fin si

$nodos\_no\_visitados\_original \leftarrow no\_visitados$

Fin para

Fin para

Si $nueva\_ruta = nulo$

agregar $ruta\_original$ a $rutas$

$ruta\_original \leftarrow inicializar\_ruta(semilla, capacidad)$

Si $nodos\_no\_visitados\_original$ está vacío

agregar $ruta\_original$ a $rutas$

Fin si

Sino

$ruta\_original \leftarrow nueva\_ruta$

$nodos\_no\_visitados\_original \leftarrow no\_visitados$

Si $nodos\_no\_visitados\_original$ está vacío

agregar $ruta\_original$ a $rutas$

Fin si

Fin si

Hasta ($nodos\_no\_visitados\_original$ esté vacío)

APÉNDICE B

MANUAL DE USUARIO DE RuGle

RuGle es un script desarrollado en Ruby, el cual, le permite al ususario determinar una solución factible para el problema multiobjetivo de enrutamiento de vehículos con ventanas de tiempo duras (VRPHTW), sujeto a rutas con congestión a determinadas horas del día, orientado a un entorno real con datos reales, a través de la heurística de inserción de Solomon e integrado con la API de Google Maps.

El presente manual, le brinda al usuario una guía útil y práctica para utilizar RuGle. En la **sección B.1**, se explica como construir las instancias de prueba para el VRPHTW a resolver. En la **sección B.2**, se dan las pautas para ejecutar el script. Y en la **sección B.3**, se explica la estructura de los resultados que arroja RuGle, al igual que los archivos que genera.

B.1. CONSTRUCCIÓN DE INSTANCIAS:

Los datos incluidos en la instancia de prueba a construir, se guardarán en una hoja de cálculo de Excel, la cual debe contar con las siguientes cabeceras:

"ID": debe ser un numero entero único que no se repite.

"ADDRESS": debe contener los datos con la ubicación del depósito central, al igual que la de cada uno de los clientes que conforman la instancia a construir, esto es: la dirección, ciudad, departamento y país.

"DEMAND": debe contener la demanda de cada uno de los clientes. Para el depósito central, la demanda será 0.

"READY __ TIME": representa la hora más temprano a la que el cliente i acepta empezar a ser atendido (formato 24 horas).

"DUE __ DATE": representa la hora más tarde hasta la que el cliente i acepta ser atendido (formato 24 horas).

"SERVICE __ TIME": debe contener el tiempo que tarda el vehículo k en servir al cliente i en su ubicación (expresado en segundos).

Para todas las instancias de prueba construidas, la flota es homogénea de máximo 25 vehículos, y cada vehículo tienen una capacidad de 200 (esto se especifica dentro del código de RuGle, línea 17 y 18).

Ejemplo:

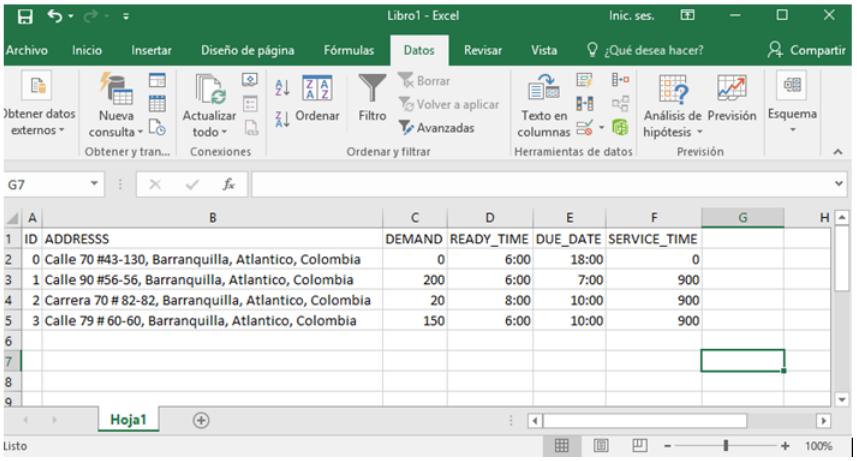

Figura B.1: Construcción de Instancias para Resolver el VRPHTW Empleando RuGle

Al terminar de registrar los datos en la hoja de cálculo de Excel, se debe guardar como un archivo ".csv".

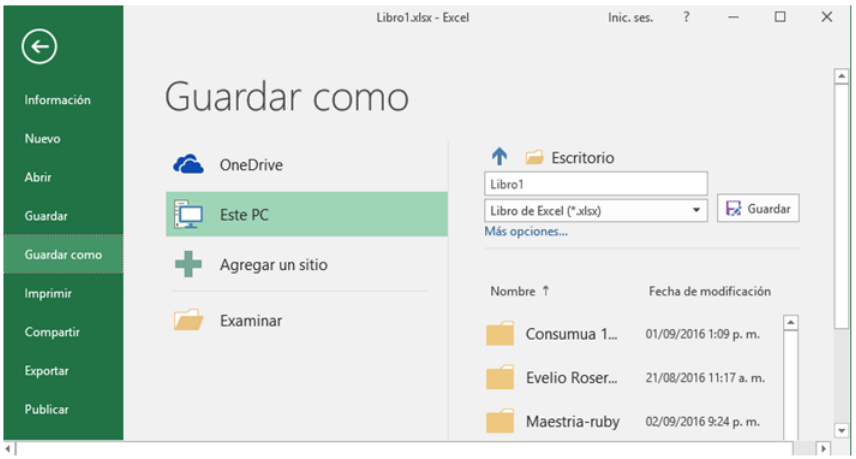

Figura B.2: Cómo Guardar el Archivo que Contiene los Datos de la Instancia Construida en Excel

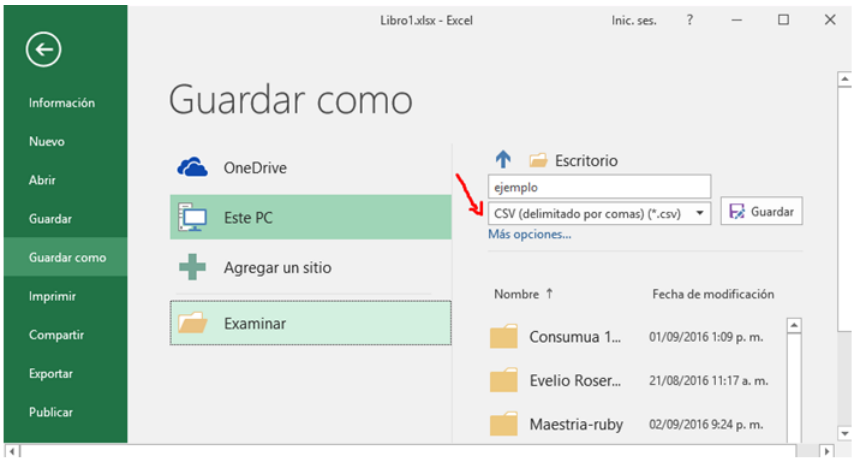

Figura B.3: Cómo Cambiar la Extensión del Archivo que Contiene los Datos de la Instancia Construida, de .xls a .csv

El archivo se debe de guardar en la carpeta "In" que se encuentra dentro del proyecto, una vez se corra el programa la nueva instancia aparecerá en el menú.

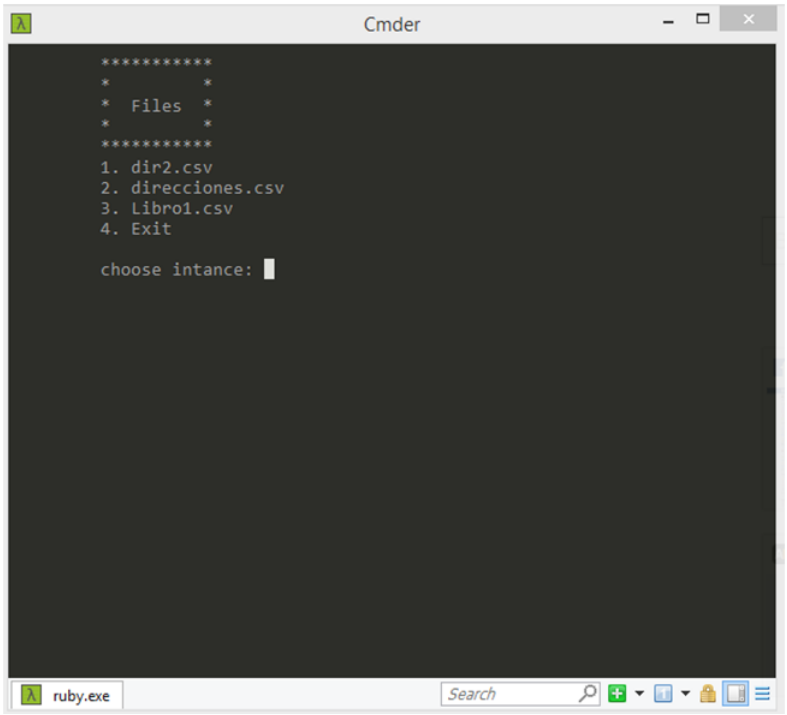

Figura B.4: El Menú Contiene la Nueva Instancia Construida

B.2. CÓMO EJECUTAR RuGle

Los usuarios de RuGle deben ingresar a la carpeta donde se encuentra el proyecto y ejecutar el siguiente comando:

ruby solomon_ google.rb

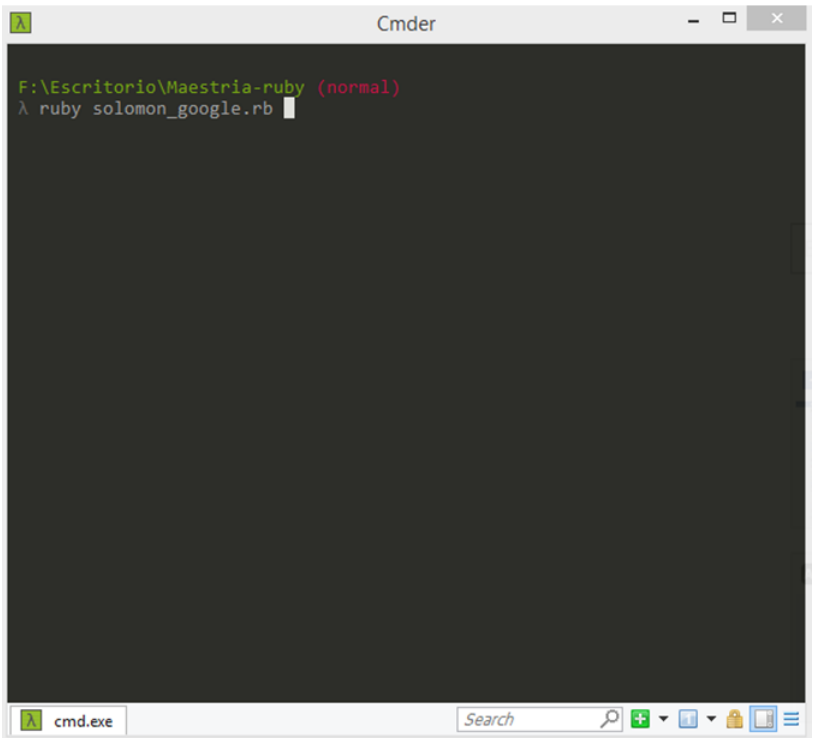

Figura B.5: Cómo Ejecutar RuGle

RuGle mostrará la lista de instancias disponibles:

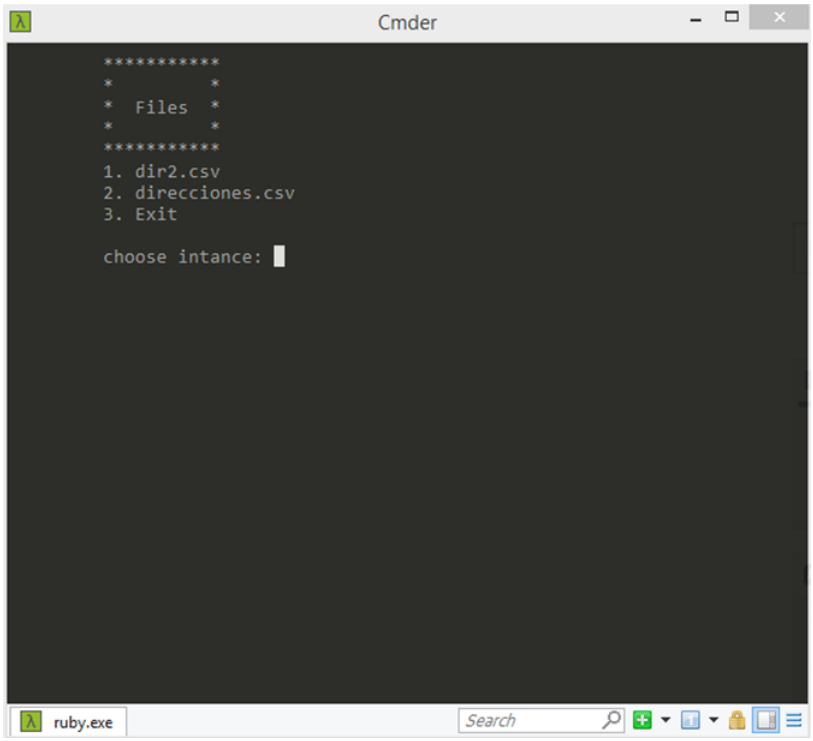

Figura B.6: Listado de RuGle

Se selecciona la instancia a correr, tal cómo indica el menú, y se esperan los resultados.

Cabe resaltar que RuGle maneja el tiempo en segundos y la distancia en metros.

B.3. RESULTADOS ARROJADOS POR RuGle

Al finalizar, el programa mostrará los siguientes datos:

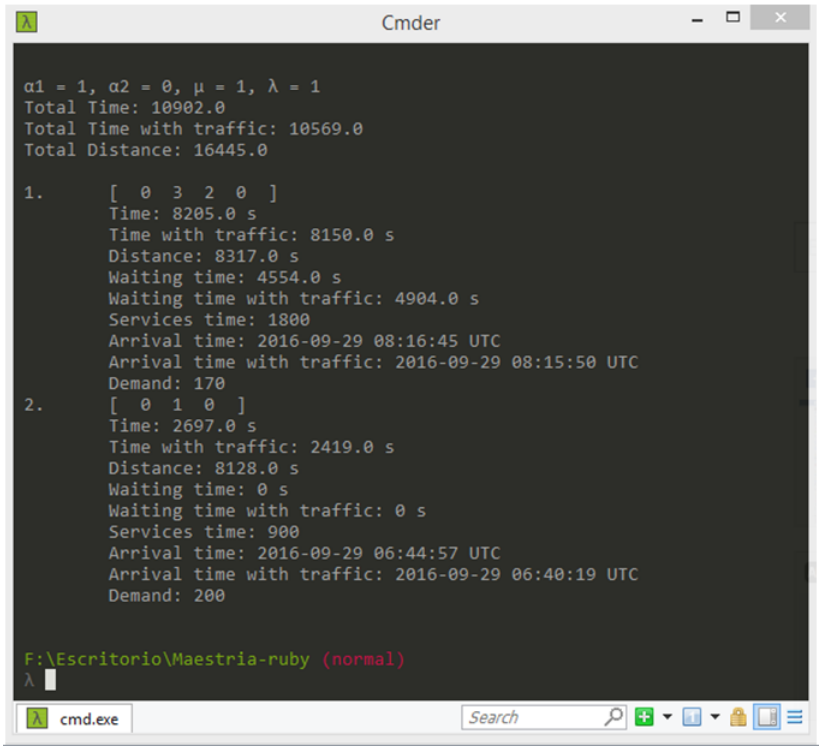

Figura B.7: Resultados Arrojados por RuGle

El tiempo total: es la sumatoria del tiempo que le toma a cada vehículo de la flota hacer el recorrido completo de su ruta sin tener en cuenta la congestión vehicular (tráfico).

El tiempo total con tráfico: es la sumatoria del tiempo que le toma a cada vehículo de la flota hacer el recorrido completo de su ruta pero teniendo en cuenta el tráfico.

La distancia total: es la sumatoria de la distancia recorrida por cada vehículo que conforma la flota.

Tamaño de la flota vehicular: es el número total de vehículos que se requieren para atender a todos los clientes.

La solución que arroja RuGle, presenta, además:

- La ruta que cada vehículo de la flota debe recorrer, iniciando y finalizando en el depósito central, con el orden en el cual los clientes deben ser visitados.
- El tiempo total que le toma a cada vehículo de la flota hacer el recorrido completo de su ruta sin tener en cuenta el tráfico.
- El tiempo total que le toma a cada vehículo de la flota hacer el recorrido completo de su ruta teniendo en cuenta el tráfico.
- La distancia total recorrida por cada vehículo que conforma la flota.
- El tiempo total de espera, de cada vehículo de la flota.
- El tiempo total de espera, teniendo en cuenta el tráfico, de cada vehículo de la flota.
- El tiempo total de servicio de cada vehículo de la flota.
- La hora de llegada al depósito central sin tener en cuenta el tráfico
- La hora de llegada al depósito teniendo en cuenta el tráfico.

Cabe resaltar que el tiempo de partida de la flota desde el depósito central, es el mismo para todos los vehículos que la conforman y se encuentra dado, es decir, especificado dentro de la instancia de prueba a correr, bajo el título de "READY _ TIME" correspondiente al nodo 0.

En la carpeta del proyecto, RuGle generará 2 archivos:

1. **out.txt:** contendrá la solución del VRPHTW. Esta solución incluye los resultados mostrados anteriormente.

2. **output.html:** mostrará gráficamente el recorrido, contenido en la solución, que cada vehículo deberá hacer. Cómo se muestra en la siguiente figura:

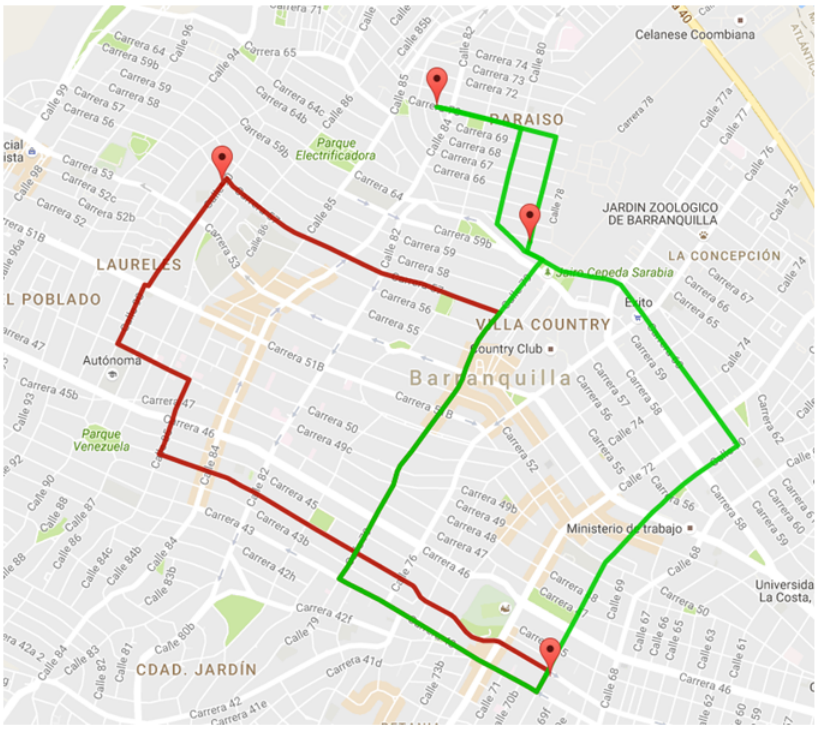

Figura B.8: Solución Gráfica Obtenida con RuGle

APÉNDICE C

MANUAL DEL SISTEMA PARA RuGle

En la **sección C.1** del presente manual, el usuario encontrará información importante acerca de los requerimientos técnicos del sistema. Y en la **sección C.2**, una descripción de los pasos para la descarga e instalación de los programas necesarios para el correcto funcionamiento de RuGle.

C.1. REQUERIMIENTOS TÉCNICOS:

Requerimientos de hardware:

- Procesador: 32 bits (x86) o 64 bits (x64) a 1 gigahercio (GHz) o más.
- Memoria RAM: 1 gigabyte (GB) (32 bits) o memoria RAM de 2 GB (64 bits).
- Espacio en disco duro: 16 GB (32 bits) o 20 GB (64 bits).
- Acceso a internet.

Requerimientos de software:

- Windows 7 o superior.
- Git.
- Ruby 2.3.3.
- Bundler
- Navegador web (de preferencia Google Chrome).

C.2. INSTALACIÓN:

Ruby:

Descargar el instalador de Ruby del siguiente enlace:

https://dl.bintray.com/oneclick/rubyinstaller/rubyinstaller-2.3.3.exe

Seguir los pasos de instalación.

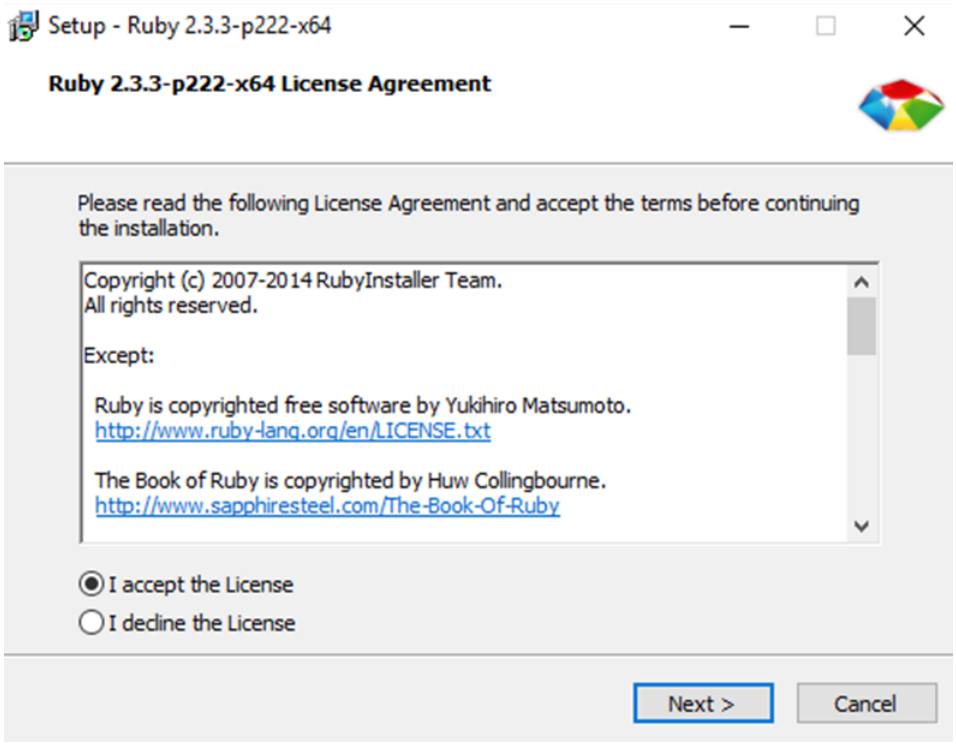

Figura C.1: Acuerdo de Licencia

Marcamos las opciones de "Add Ruby executables to your PATH" y "Associate .rb and .rbw file with Ruby installation"

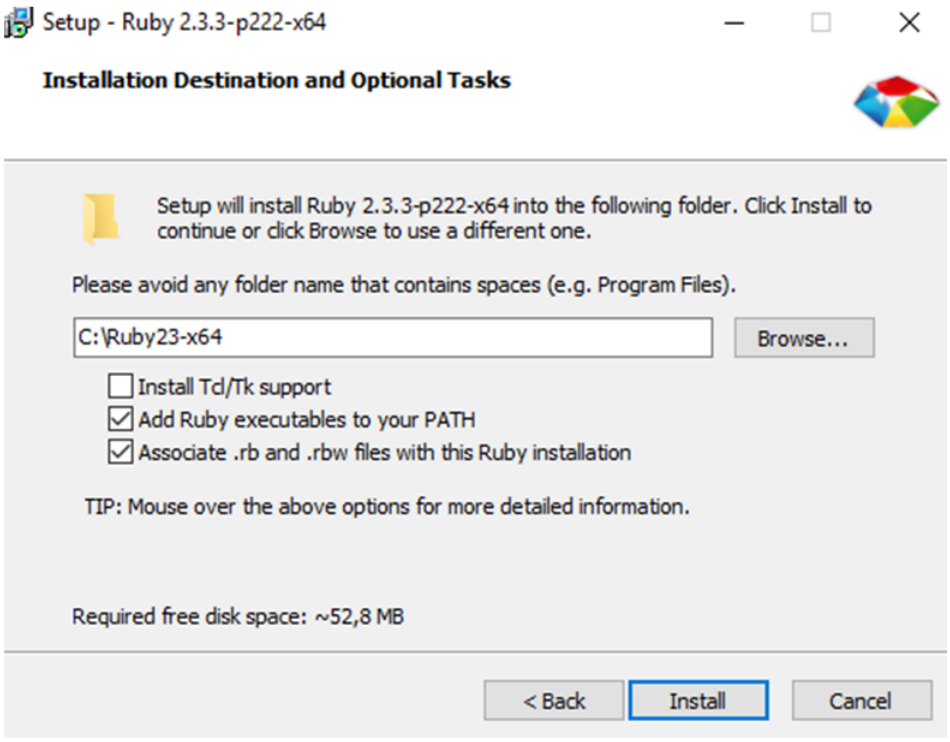

Figura C.2: Destino de la Instalación

Presionamos el botón Install para finalizar con la instalación de Ruby.

Git:

Descargar el instalador de git del siguiente enlace:

https://git-scm.com/download/win

Figura C.3: Información General

Solo seguimos la guía del instalador, presionando Next en cada caso y luego el botón finalizar la instalación.

Bundler:

Abrimos la consola de comandos de Windows ya sea buscado "cmd" en el menú de inicio o presionado las teclas Windows + R simultáneamente, digitar "cmd" y así abrir la consola de comando.

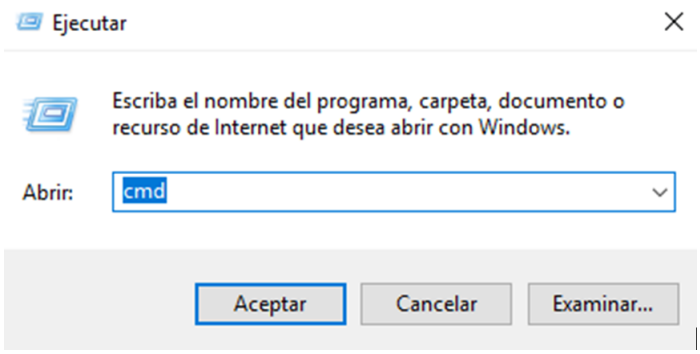

Figura C.4: Ventana del comando "Ejecutar"

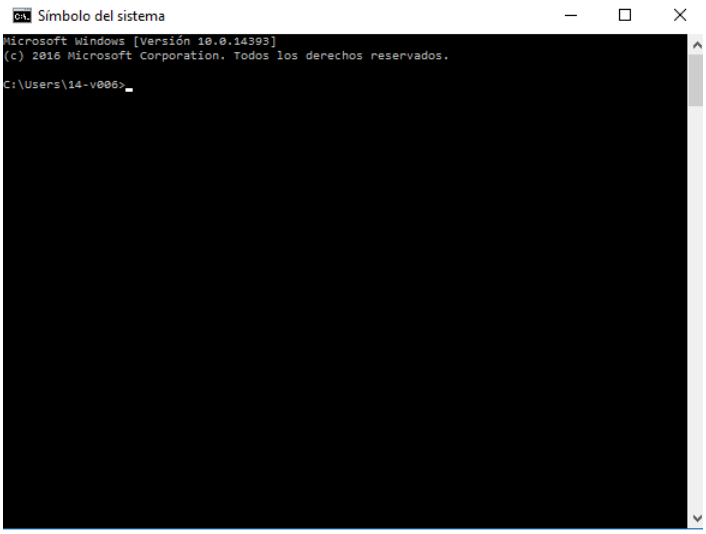

Figura C.5: Consola cmd

Digitamos el siguiente comando en la ventana de cmd:

gem install bundler

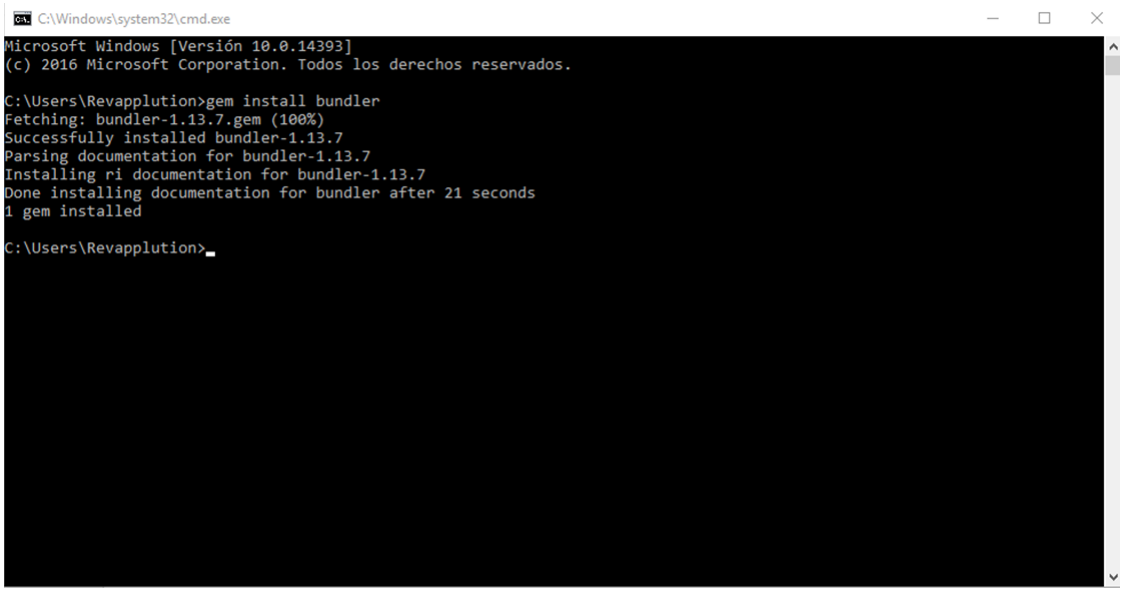

Figura C.6: Gem install bundler

En este punto ya se tiene todo lo necesario para ejecutar RuGle. Cabe resaltar que RuGle no es un programa, es un script que se ejecuta, por lo cual este no requiere instalación.

Para descargar y ejecutar RuGle, debemos digitar el siguiente comando en la ventana cmd:

git clone https://github.com/Andoresu250/RuGle

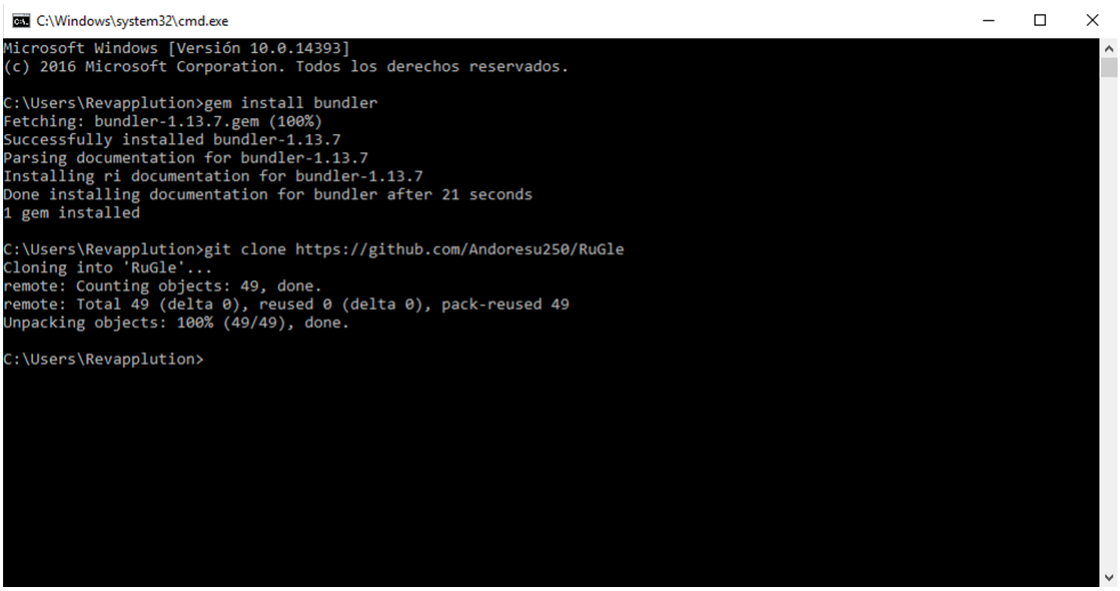

Figura C.7: Cómo clonar el repositorio

Una vez clonado el repositorio digitamos el siguiente comando:

```
git checkout trafico
```

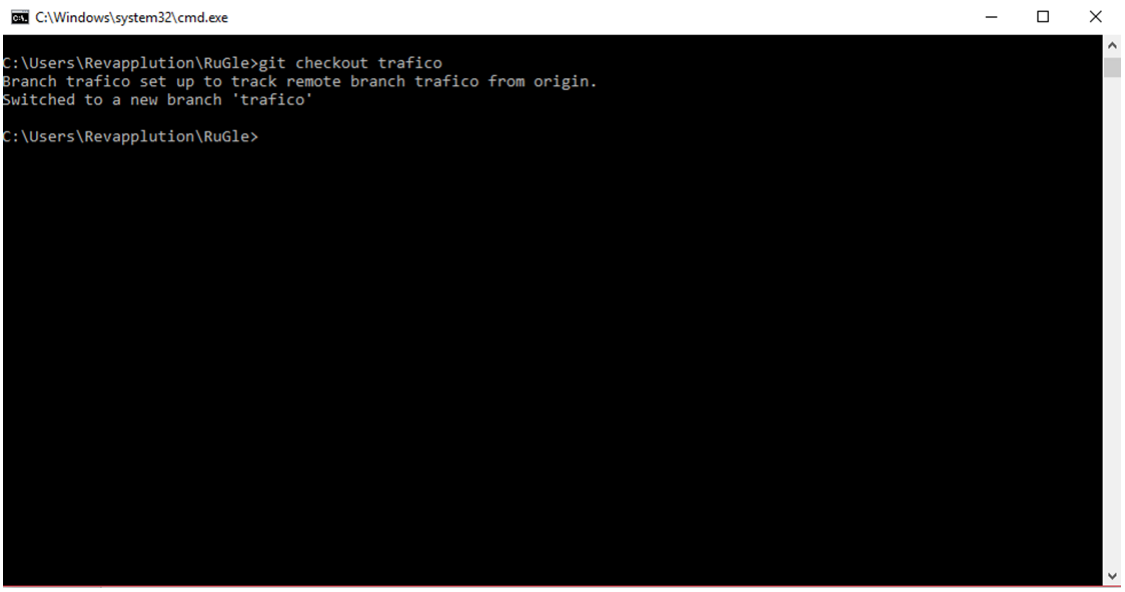

Figura C.8: Cómo ejecutar el comando git checkout trafico

Para ingresar al directorio de Rugle se ejecuta el siguiente comando:

```
cd RuGle
```

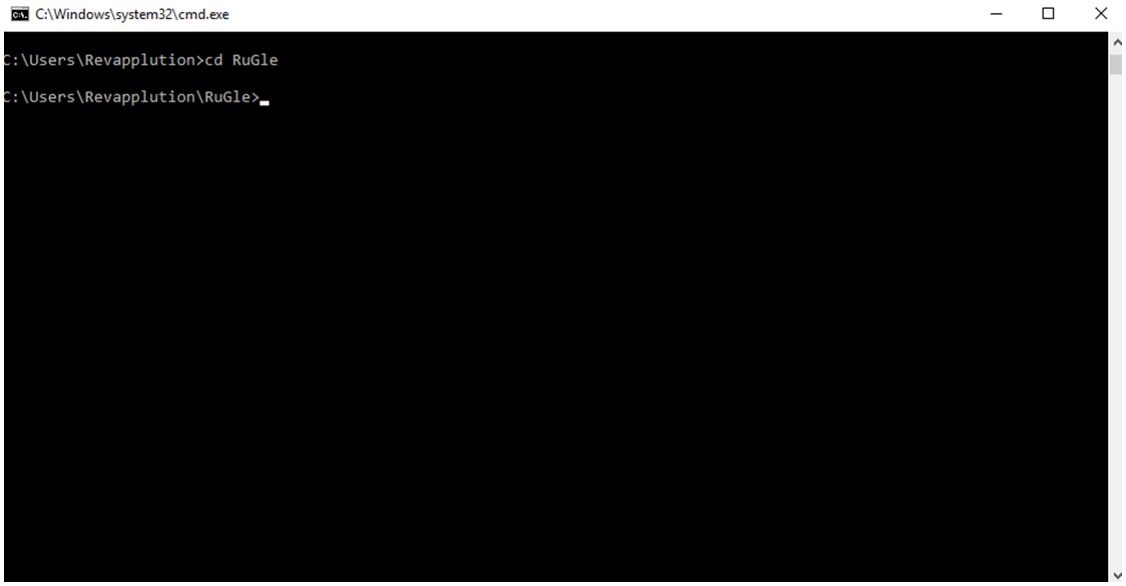

Figura C.9: Cómo ingresar al directorio de RuGle

Una vez aquí, instalamos todas las dependecias de Rugle con el comando:

```
bundle install
```

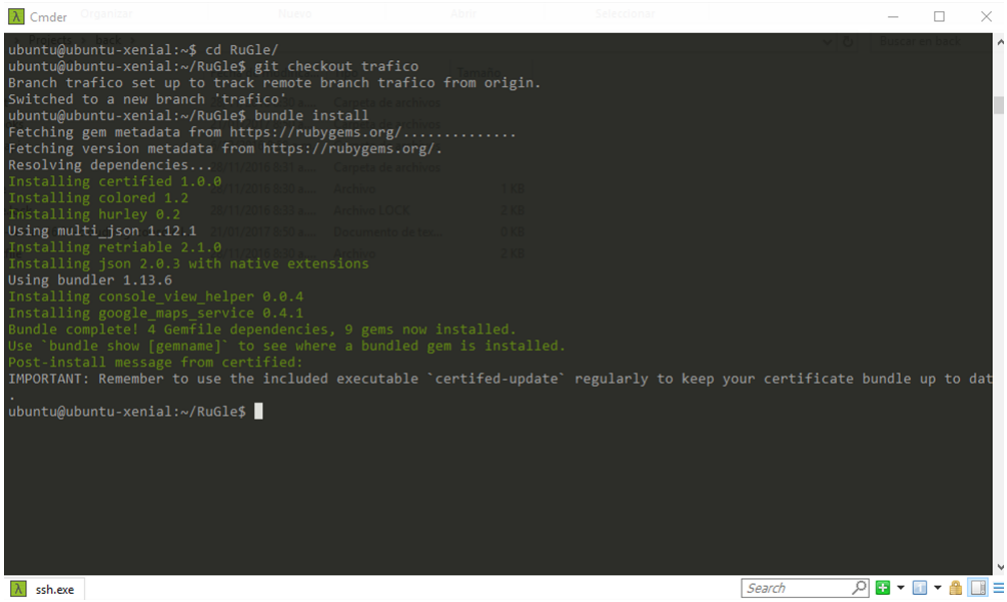

Figura C.10: Cómo instalar las dependencias de RuGle

Y por último ejecutamos el comando:

ruby solomon_google.rb

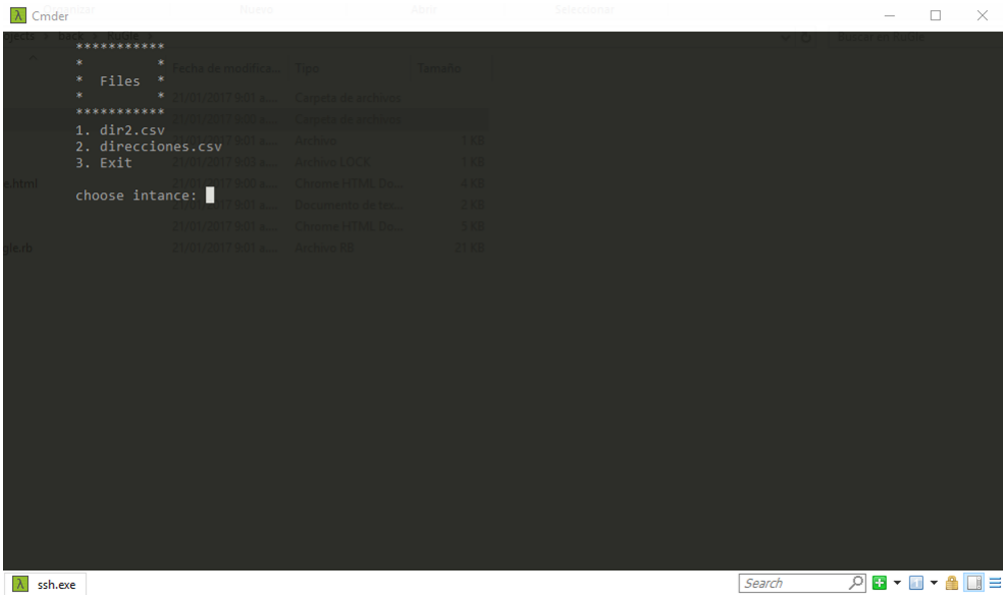

Figura C.11: Cómo ejecutar RuGle